```
package org.deta.tinos.array;
import org.json.XML;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import org.deta.tinos.stable.Stable;
import org.json.JSONObject;
import com.google.gson.Gson;
public class ArraySwap{
    public static String arrayToJson(Gson gson, Object[] object) {
        return gson.toJson(object);
    }
    public static String arrayToXml(Gson gson, Object[] object) {
        return XML.toString(new JSONObject(gson.toJson(object)));
    }
    public static Map<String, Object> arrayToMap(Gson gson
            , Object[] objects) {
        Map<String, Object> map= new HashMap<>();
        int i= 0;
        for(Object object: objects) {
            map.put(Stable.STRING_EMPTY+ i++, object);
        }
        return map;
    }
    public static Vector<Object> arrayToVector(Gson gson
            , Object[] objects) {
        Vector<Object> vector= new Vector<>();
        for(Object object: objects) {
            vector.add(object);
        }
        return vector;
    }
    public static List<Object> arrayToList(Object[] objects) {
        List<Object> list= new ArrayList<>();
        for(Object object: objects) {
            list.add(object);
        }
        return list;
    }
    public static Iterator<Object> arrayToIterator(Object[] objects) {
        List<Object> list= new ArrayList<>();
        for(Object object: objects) {
```

```
            list.add(object);
        }
        return list.iterator();
    }
    public static Set<Object> arrayToSet(Object[] objects) {
        Set<Object> sets= new TreeSet<>();
        for(Object object: objects) {
            sets.add(object);
        }
        return sets;
    }
    public static Object[][] arrayToMatrix(Object[] objects, int widthRange){
        Object[][] output= new Object[widthRange][objects.length/ widthRange];
        for(int i= 0; i< widthRange; i++) {
            for(int j= 0; j< objects.length/ widthRange; j++) {
                output[i][j]= objects[i* widthRange+ j];
            }
        }
        return output;
    }
    public static Object[][][] arrayToMatrix3D(Object[] objects, int widthRange
            , int heightRange){
        Object[][][] output
        = new Object[widthRange][heightRange][objects.length/ widthRange
                                              / heightRange];
        for(int i= 0; i< widthRange; i++) {
            for(int j= 0; j< heightRange; j++) {
                for(int k= 0; k< objects.length/ widthRange/ heightRange; k++) {
                    output[i][j][k]= objects[i* widthRange* heightRange
                                             + j* heightRange+ k];
                }
            }
        }
        return output;
    }
}
```

```
package org.deta.tinos.array;
public class ArrayValidation{
    //以后 validation大体包括 check和fix， 2个部分。
    public static boolean arrayIntCheck(int[] array
            , int min, int max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max|| array[i]< min) {
                return false;
            }
        }
```

```
        return true;
    }
    public static int[] arrayIntFix(int[] array
            , int min, int max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max) {
                array[i]= max;
            }
            if(array[i]< min) {
                array[i]= min;
            }
        }
        return array;
    }
    public static boolean arrayLongCheck(long[] array
            , long min, long max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max|| array[i]< min) {
                return false;
            }
        }
        return true;
    }
    public static long[] arrayLongFix(long[] array
            , long min, long max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max) {
                array[i]= max;
            }
            if(array[i]< min) {
                array[i]= min;
            }
        }
        return array;
    }
    public static boolean arrayDoubleCheck(double[] array
            , double min, double max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max|| array[i]< min) {
                return false;
            }
        }
        return true;
    }
    public static double[] arrayDoubleFix(double[] array
            , double min, double max) {
        for(int i= 0; i< array.length; i++) {
```

```
            if(array[i]> max) {
                array[i]= max;
            }
            if(array[i]< min) {
                array[i]= min;
            }
        }
        return array;
    }
    public static boolean arrayFloatCheck(float[] array
            , float min, float max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max|| array[i]< min) {
                return false;
            }
        }
        return true;
    }
    public static float[] arrayFloatFix(float[] array
            , float min, float max) {
        for(int i= 0; i< array.length; i++) {
            if(array[i]> max) {
                array[i]= max;
            }
            if(array[i]< min) {
                array[i]= min;
            }
        }
        return array;
    }
}
```

```
package org.deta.tinos.csv;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFDateUtil;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;
@SuppressWarnings("static-access")
public class CSVSwap{
    public static Object[][] xlsOrCsvFileToObjectMartix(String filePath
```

```
        , int pageSheetIndex) throws IOException {
    FileInputStream fileInputStream= new FileInputStream(filePath);
    POIFSFileSystem pOIFSFileSystem= new POIFSFileSystem(fileInputStream);
    HSSFWorkbook hSSFWorkbook= new HSSFWorkbook(pOIFSFileSystem);
    HSSFSheet hSSFSheet= hSSFWorkbook.getSheetAt(Integer
            .valueOf(pageSheetIndex));
    HSSFRow hSSFRow= hSSFSheet.getRow(0);
    Object[][] output= new String[hSSFSheet.getPhysicalNumberOfRows()]
            [hSSFRow.getLastCellNum()];
    for (int i= 0; i< hSSFSheet.getPhysicalNumberOfRows(); i++) {//ROW
        if (null!= (hSSFRow= hSSFSheet.getRow(i))) {
            for(int j= 0; j< hSSFRow.getLastCellNum(); j++){//CULUMN
                HSSFCell hSSFCell= hSSFRow.getCell(j);
                if(hSSFCell.CELL_TYPE_STRING== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getStringCellValue();
                }
                if(hSSFCell.CELL_TYPE_BOOLEAN== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getBooleanCellValue();
                }
                if(hSSFCell.CELL_TYPE_NUMERIC== hSSFCell.getCellType()){
                    if(HSSFDateUtil.isCellDateFormatted(hSSFCell)){
                        output[i][j]= hSSFCell.getDateCellValue();
                    }else{
                        output[i][j]= hSSFCell.getNumericCellValue();
                    }
                }
                if(hSSFCell.CELL_TYPE_ERROR== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getErrorCellValue();
                }
            }
        }
    }
    return output;
}
public static List<Object[]> xlsOrCsvFileToListObject(String filePath
        , int pageSheetIndex) throws IOException {
    FileInputStream fileInputStream= new FileInputStream(filePath);
    POIFSFileSystem pOIFSFileSystem= new POIFSFileSystem(fileInputStream);
    HSSFWorkbook hSSFWorkbook= new HSSFWorkbook(pOIFSFileSystem);
    HSSFSheet hSSFSheet= hSSFWorkbook.getSheetAt(Integer
            .valueOf(pageSheetIndex));
    HSSFRow hSSFRow= hSSFSheet.getRow(0);
    List<Object[]> list= new ArrayList<>();
    for (int i= 0; i< hSSFSheet.getPhysicalNumberOfRows(); i++) {//ROW
        Object[] objectRow= new String[hSSFRow.getLastCellNum()];
        if (null!= (hSSFRow= hSSFSheet.getRow(i))) {
            for(int j= 0; j< hSSFRow.getLastCellNum(); j++){//CULUMN
```

```
                HSSFCell hSSFCell= hSSFRow.getCell(j);
                if(hSSFCell.CELL_TYPE_STRING== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getStringCellValue();
                }
                if(hSSFCell.CELL_TYPE_BOOLEAN== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getBooleanCellValue();
                }
                if(hSSFCell.CELL_TYPE_NUMERIC== hSSFCell.getCellType()){
                    if(HSSFDateUtil.isCellDateFormatted(hSSFCell)){
                        objectRow[j]= hSSFCell.getDateCellValue();
                    }else{
                        objectRow[j]= hSSFCell.getNumericCellValue();
                    }
                }
                if(hSSFCell.CELL_TYPE_ERROR== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getErrorCellValue();
                }
            }
        }
        list.add(objectRow);
    }
    return list;
}
public static List<Object[]> xlsOrCsvFileToRangedListObject(String filePath
        , int pageSheetIndex, Map<Integer, Boolean> rows
        , Map<Integer, Boolean> culumns) throws IOException {
    FileInputStream fileInputStream= new FileInputStream(filePath);
    POIFSFileSystem pOIFSFileSystem= new POIFSFileSystem(fileInputStream);
    HSSFWorkbook hSSFWorkbook= new HSSFWorkbook(pOIFSFileSystem);
    HSSFSheet hSSFSheet= hSSFWorkbook
            .getSheetAt(Integer.valueOf(pageSheetIndex));
    HSSFRow hSSFRow= hSSFSheet.getRow(0);
    List<Object[]> list= new ArrayList<>();
    for (int i= 0; i< hSSFSheet.getPhysicalNumberOfRows()
            && rows.containsKey(i); i++) {//ROW
        Object[] objectRow= new String[hSSFRow.getLastCellNum()];
        if (null!= (hSSFRow= hSSFSheet.getRow(i))) {
            for(int j= 0; j< hSSFRow.getLastCellNum()
                    && culumns.containsKey(j); j++){//CULUMN
                HSSFCell hSSFCell= hSSFRow.getCell(j);
                if(hSSFCell.CELL_TYPE_STRING== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getStringCellValue();
                }
                if(hSSFCell.CELL_TYPE_BOOLEAN== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getBooleanCellValue();
                }
                if(hSSFCell.CELL_TYPE_NUMERIC== hSSFCell.getCellType()){
```

```
                    if(HSSFDateUtil.isCellDateFormatted(hSSFCell)){
                        objectRow[j]= hSSFCell.getDateCellValue();
                    }else{
                        objectRow[j]= hSSFCell.getNumericCellValue();
                    }
                }
                if(hSSFCell.CELL_TYPE_ERROR== hSSFCell.getCellType()){
                    objectRow[j]= hSSFCell.getErrorCellValue();
                }
            }
        }
        list.add(objectRow);
    }
    return list;
}
public static Object[][] xlsOrCsvFileToRangedObjectMartix(String filePath
        , int pageSheetIndex, Map<Integer, Boolean> rows
        , Map<Integer, Boolean> culumns) throws IOException {
    FileInputStream fileInputStream= new FileInputStream(filePath);
    POIFSFileSystem pOIFSFileSystem= new POIFSFileSystem(fileInputStream);
    HSSFWorkbook hSSFWorkbook= new HSSFWorkbook(pOIFSFileSystem);
    HSSFSheet hSSFSheet= hSSFWorkbook
            .getSheetAt(Integer.valueOf(pageSheetIndex));
    HSSFRow hSSFRow= hSSFSheet.getRow(0);
    Object[][] output= new String[hSSFSheet.getPhysicalNumberOfRows()]
            [hSSFRow.getLastCellNum()];
    for (int i= 0; i< hSSFSheet.getPhysicalNumberOfRows()
            && rows.containsKey(i); i++) {//ROW
        if (null!= (hSSFRow= hSSFSheet.getRow(i))) {
            for(int j= 0; j< hSSFRow.getLastCellNum()
                    && culumns.containsKey(j); j++){//CULUMN
                HSSFCell hSSFCell= hSSFRow.getCell(j);
                if(hSSFCell.CELL_TYPE_STRING== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getStringCellValue();
                }
                if(hSSFCell.CELL_TYPE_BOOLEAN== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getBooleanCellValue();
                }
                if(hSSFCell.CELL_TYPE_NUMERIC== hSSFCell.getCellType()){
                    if(HSSFDateUtil.isCellDateFormatted(hSSFCell)){
                        output[i][j]= hSSFCell.getDateCellValue();
                    }else{
                        output[i][j]= hSSFCell.getNumericCellValue();
                    }
                }
                if(hSSFCell.CELL_TYPE_ERROR== hSSFCell.getCellType()){
                    output[i][j]= hSSFCell.getErrorCellValue();
```

```
                    }
                }
            }
        }
        return output;
    }
}
```

```
package org.deta.tinos.date;
import java.sql.Timestamp;
import java.util.Date;
import org.deta.tinos.stable.Stable;
@SuppressWarnings({Stable.SUPPRESS_WARNINGS_DEPRECATION
    , Stable.SUPPRESS_WARNINGS_STATIC_ACCESS})
public class DateSwap{
    public static String dateToGMTString(Date date) {
        return date.toGMTString();
    }
    public static String dateToYYYYMMDD(Date date) {
        return Stable.STRING_EMPTY+ date.getYear()
        + Stable.STRING_PER+ date.getMonth()
        + Stable.STRING_PER+ date.getDay();
    }
    public static String dateToHHMMSS(Date date) {
        return Stable.STRING_EMPTY+ date.getHours()
        + Stable.STRING_PER+ date.getMinutes()
        + Stable.STRING_PER+ date.getSeconds();
    }
    public static String dateToMiliSeconds(Date date) {
        return Stable.STRING_EMPTY+ date.getTime();
    }
    public static String getCurrentMiliSeconds(Date date) {
        return Stable.STRING_EMPTY+ new Date().getTime();
    }
    public static Timestamp dateToTimeStamp(Date date) {
        long utc= date.UTC(date.getYear(), date.getMonth(), date.getDay()
                , date.getHours(), date.getMinutes(), date.getSeconds());
        return new Timestamp(utc);
    }
    public static long timeStampToMiliSeconds(Timestamp timestamp) {
        return timestamp.getTime();
    }
    public static String timeStampToMiliSecondsWithSize(Timestamp timestamp
            , int size) {
        String time= Stable.STRING_EMPTY+ timestamp.getTime();
        return time.substring(0, size);
    }
```

```
    public static String dateStringToMiliSeconds(String string) {
        return Stable.STRING_EMPTY+ new Date(string).getTime();
    }
}
```

```
package org.deta.tinos.date;
import java.io.IOException;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;
import org.deta.tinos.stable.Stable;
//refer www.bjtime.com 第 12，13，14 行 共三行
//注意如果 bjtime 网站关闭了这个功能，该函数将无效，请再互联网搜索其他提供时间校验的网址。
public class DateValidation{
    public static boolean currentSystemTimeCheck(long offsetUTC8
            , long miliSecondsDistinction) throws IOException {
        URL url= new URL(Stable.STRING_BJTIME);
        URLConnection uRLConnection= url.openConnection();
        uRLConnection.connect();
        long bjtime= uRLConnection.getDate();
        long system= currentSystemTimeToUTC(offsetUTC8);
        if(miliSecondsDistinction< Math.abs(bjtime- system)) {
            return false;
        }
        return true;
    }
    public static long currentSystemTimeToUTC(long offsetUTC8) {
        return new Date().getTime()+ offsetUTC8;
    }
}
```

```
package org.deta.tinos.hash;
import com.google.gson.Gson;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;
import org.json.JSONObject;
import org.json.XML;
public class HashSwap{
    public static String hashTableToJson(Gson gson
            , Hashtable<String, Object> hashtable) {
        return gson.toJson(hashtable);
    }
    public static String hashTableToXml(Gson gson
            , Hashtable<String, Object> hashtable){
        JSONObject jSONObject= new JSONObject(gson.toJson(hashtable));
```

```
        return XML.toString(jSONObject);
    }
    public static List<Object> hashTableToList(Gson gson
            , Hashtable<String, Object> hashtable){
        List<Object> list= new ArrayList<>();
        Iterator<String> iterator= hashtable.keySet().iterator();
        while(iterator.hasNext()) {
            list.add(hashtable.get(iterator.next()));
        }
        return list;
    }
    public static Object[] hashTableToObjectArray(Gson gson
            , Hashtable<String, Object> hashtable){
        List<Object> list= new ArrayList<>();
        Iterator<String> iterator= hashtable.keySet().iterator();
        while(iterator.hasNext()) {
            list.add(hashtable.get(iterator.next()));
        }
        return list.toArray();
    }
}
```

```
package org.deta.tinos.http;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import org.deta.tinos.stable.Stable;
public class HttpUnicode{
    //这个函数在作者去年开发股市分析软件的时候从雪球等网站采样抓数据，发现有 GBK
    //utf-8 gb2312 ascii 等格式数据比较混乱，于是进行统一格式处理。
    public String getJson(String urlString, String jsonString)
            throws IOException{
        String code= Stable.STRING_EMPTY;
        URL url= new URL(urlString);
        HttpURLConnection connection= (HttpURLConnection)url.openConnection();
        connection.setRequestMethod(Stable.HTTP_GET);
        connection
        .setRequestProperty(Stable.CONTENT_TYPE, Stable.APPLICATION_JSON_UTF8);
        connection.setDoOutput(true);
        connection.setInstanceFollowRedirects(false);
        StringBuffer sbuffer= new StringBuffer();
        sbuffer.append(jsonString);
        OutputStream os= connection.getOutputStream();
```

```
        os.write(sbuffer.toString().getBytes());
        os.flush();
        StringBuilder stringBuilder= new StringBuilder();
        BufferedReader bufferedReader= null;
        try {
            InputStream inputStream= connection.getInputStream();
            int caherset_size= connection.getHeaderFields().size();
            for(int i= 0; i< caherset_size; i++) {
                String temp= connection.getHeaderField(i);
                if(temp.contains(Stable.CHARSET)) {
                    if(temp.toUpperCase().contains(Stable.CHARSET_GBK)) {
                        code= Stable.CHARSET_GBK;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_UTF_8)
                            || temp.toUpperCase().contains(Stable.CHARSET_UTF8)) {
                        code= Stable.CHARSET_UTF_8;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_GB2312)) {
                        code= Stable.CHARSET_GB2312;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_ASCII)) {
                        code= Stable.CHARSET_ASCII;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_UNICODE)) {
                        code= Stable.CHARSET_UNICODE;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_ISO_8859_1)) {
                        code= Stable.CHARSET_ISO_8859_1;
                    }
                }
            }
            if (null!= inputStream) {
                bufferedReader
                =  new BufferedReader(new InputStreamReader(inputStream, code));
                String lines;
                while ((lines= bufferedReader.readLine()) != null) {
                    stringBuilder.append(lines);
                }
            } else {
                stringBuilder.append(Stable.STRING_EMPTY);
            }
        } catch (IOException ex) {
            //throw ex;
        } finally {
            if (null!= bufferedReader) {
                try {
                    bufferedReader.close();
```

```
                } catch (IOException ex) {
                    //throw ex;
                }
            }
        }
        return new String(stringBuilder.toString().getBytes(), Stable.CHARSET_UTF_8);
    }
    public String postXML(String urlString,String XMLString) throws IOException{
        URL url= new URL(urlString);
        HttpURLConnection connection= (HttpURLConnection)url.openConnection();
        connection.setRequestMethod(Stable.HTTP_POST);
        connection.setRequestProperty(Stable.CONTENT_TYPE, Stable.APPLICATION_XML);
        connection.setDoOutput(true);
        connection.setInstanceFollowRedirects(false);
        StringBuffer stringBuffer= new StringBuffer();
        stringBuffer.append(XMLString);
        OutputStream outputStream= connection.getOutputStream();
        outputStream.write(stringBuffer.toString().getBytes());
        outputStream.flush();
        String requestBody= Stable.STRING_EMPTY;
        StringBuilder stringBuilder= new StringBuilder();
        BufferedReader bufferedReader= null;
        try {
            InputStream inputStream= connection.getInputStream();
            if (null!= inputStream) {
                bufferedReader
                = new BufferedReader(new InputStreamReader(inputStream));
                char[] charBuffer= new char[128];
                int bytesRead= -1;
                while ((bytesRead= bufferedReader.read(charBuffer)) > 0) {
                    stringBuilder.append(charBuffer, 0, bytesRead);
                }
            } else {
                stringBuilder.append(Stable.STRING_EMPTY);
            }
        } catch (IOException ex) {
            throw ex;
        } finally {
            if (null!= bufferedReader) {
                try {
                    bufferedReader.close();
                } catch (IOException ex) {
                    throw ex;
                }
            }
        }
        requestBody= stringBuilder.toString();
```

```
        return requestBody;
    }
    public String postJsonWithSercurity(String urlString
            , String jsonString) throws IOException{
        URL url= new URL(urlString);
        HttpURLConnection connection= (HttpURLConnection)url.openConnection();
        connection.setRequestMethod(Stable.HTTP_POST);
        connection.setRequestProperty(Stable.CONTENT_TYPE, Stable.APPLICATION_JSON);
        connection.setDoOutput(true);
        connection.setInstanceFollowRedirects(false);
        StringBuffer sbuffer= new StringBuffer();
        sbuffer.append(jsonString);
        OutputStream os= connection.getOutputStream();
        os.write(sbuffer.toString().getBytes());
        os.flush();
        String requestBody= Stable.STRING_EMPTY;
        StringBuilder stringBuilder= new StringBuilder();
        BufferedReader bufferedReader= null;
        try {
            InputStream inputStream= connection.getInputStream();
            if (null!= inputStream) {
                bufferedReader
                = new BufferedReader(new InputStreamReader(inputStream));
                char[] charBuffer= new char[128];
                int bytesRead= -1;
                while ((bytesRead= bufferedReader.read(charBuffer)) > 0) {
                    stringBuilder.append(charBuffer, 0, bytesRead);
                }
            } else {
                stringBuilder.append(Stable.STRING_EMPTY);
            }
        } catch (IOException ex) {
            throw ex;
        } finally {
            if (null!= bufferedReader) {
                try {
                    bufferedReader.close();
                } catch (IOException ex) {
                    throw ex;
                }
            }
        }
        requestBody= stringBuilder.toString();
        return requestBody;
    }
    public String postXMLWithSercurity(String urlString, String XMLString)
            throws IOException{
```

```
        URL url= new URL(urlString);
        HttpURLConnection connection= (HttpURLConnection)url.openConnection();
        connection.setRequestMethod(Stable.HTTP_POST);
        connection.setRequestProperty(Stable.CONTENT_TYPE, Stable.APPLICATION_XML);
        connection.setDoOutput(true);
        connection.setInstanceFollowRedirects(false);
        StringBuffer sbuffer= new StringBuffer();
        sbuffer.append(XMLString);
        OutputStream os= connection.getOutputStream();
        os.write(sbuffer.toString().getBytes());
        os.flush();
        String requestBody= Stable.STRING_EMPTY;
        StringBuilder stringBuilder= new StringBuilder();
        BufferedReader bufferedReader= null;
        try {
            InputStream inputStream= connection.getInputStream();
            if (null!= inputStream) {
                bufferedReader= new BufferedReader
                        (new InputStreamReader(inputStream));
                char[] charBuffer= new char[128];
                int bytesRead= -1;
                while ((bytesRead= bufferedReader.read(charBuffer)) > 0) {
                    stringBuilder.append(charBuffer, 0, bytesRead);
                }
            } else {
                stringBuilder.append(Stable.STRING_EMPTY);
            }
        } catch (IOException ex) {
            throw ex;
        } finally {
            if (null!= bufferedReader) {
                try {
                    bufferedReader.close();
                } catch (IOException ex) {
                    throw ex;
                }
            }
        }
        requestBody= stringBuilder.toString();
        return requestBody;
    }
    public String getStatusFromString(String response) {
        for(int i=0;i<response.length();i++){
        }
        return null;
    }
    public String getHTML(String urlString, Object object) throws IOException {
```

```
        URL url= new URL(urlString);
        HttpURLConnection connection= (HttpURLConnection)url.openConnection();
        connection.setRequestMethod(Stable.HTTP_GET);
        connection.setDoOutput(true);
        connection.setInstanceFollowRedirects(false);
        StringBuilder stringBuilder= new StringBuilder();
        BufferedReader bufferedReader= null;
        String code= Stable.CHARSET_GB2312;
        try {
            InputStream inputStream= connection.getInputStream();
            int caherset_size= connection.getHeaderFields().size();
            for(int i= 0; i< caherset_size; i++) {
                String temp= connection.getHeaderField(i);
                if(temp.contains(Stable.CHARSET)|| temp.contains(Stable.TYPE) ) {
                    if(temp.toUpperCase().contains(Stable.CHARSET_GBK)) {
                        code= Stable.CHARSET_GBK;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_UTF_8)
                            || temp.toUpperCase().contains(Stable.CHARSET_UTF8)) {
                        code= Stable.CHARSET_UTF_8;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_GB2312)) {
                        code= Stable.CHARSET_GB2312;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_ASCII)) {
                        code= Stable.CHARSET_ASCII;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_UNICODE)) {
                        code= Stable.CHARSET_UNICODE;
                    }
                    if(temp.toUpperCase().contains(Stable.CHARSET_ISO_8859_1)) {
                        code= Stable.CHARSET_ISO_8859_1;
                    }
                }
            }
            if (null!= inputStream) {
                bufferedReader= new BufferedReader(new InputStreamReader(inputStream,code));
                char[] charBuffer= new char[128];
                int bytesRead= -1;
                while ((bytesRead= bufferedReader.read(charBuffer)) > 0) {
                    stringBuilder.append(charBuffer, 0, bytesRead);
                }
            } else {
                stringBuilder.append(Stable.STRING_EMPTY);
            }
        } catch (IOException ex) {
        } finally {
```

```
            if (null!= bufferedReader) {
                try {
                    bufferedReader.close();
                } catch (IOException ex) {
                }
            }
        }
        String out= new String(stringBuilder.toString().getBytes(), Stable.CHARSET_UTF_8);
        return out;
    }
}
```

```
package org.deta.tinos.image;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.awt.Image;
import java.awt.image.BufferedImage;
public class ImageSwap{
    public static void pixTableToFile(String pngOutputPath, int[][] pix
            , String fileType) throws IOException{
        BufferedImage bufferedImage= new BufferedImage(pix[0].length
                , pix.length, BufferedImage.TYPE_INT_RGB);
        for (int i= 0; i< bufferedImage.getHeight(); ++i) {
            for (int j= 0; j< bufferedImage.getWidth(); ++j) {
                bufferedImage.setRGB(j, i, (pix[i][j]<< 16)
                        | (pix[i][j]<< 8)| (pix[i][j]));
            }
        }
        ImageIO.write(bufferedImage, fileType, new File(pngOutputPath));
    }
    public static void pixRGBTableToFile(String pngOutputPath, int[][] pixRed
            , int[][] pixGreen, int[][] pixBlue, String fileType)
                    throws IOException{
        BufferedImage bufferedImage= new BufferedImage(pixRed[0].length
                , pixRed.length, BufferedImage.TYPE_INT_RGB);
        for (int i= 0; i< bufferedImage.getHeight(); ++i) {
            for (int j= 0; j< bufferedImage.getWidth(); ++j) {
                bufferedImage.setRGB(j, i, (pixRed[i][j]<< 16)
                        | (pixGreen[i][j]<< 8)| (pixBlue[i][j]));
            }
        }
        ImageIO.write(bufferedImage, fileType, new File(pngOutputPath));
    }
    public static void bufferedImageToFile(String pngOutputPath
            , BufferedImage bufferedImage, String fileType) throws IOException{
        ImageIO.write(bufferedImage, fileType, new File(pngOutputPath));
```

```
    }
    public static void imageToFile(String pngOutputPath, Image image
            , String fileType)throws IOException{
        ImageIO.write((BufferedImage)image, fileType, new File(pngOutputPath));
    }
    public static void bufferedImageToScaleImageFile(String pngOutputPath
            , BufferedImage bufferedImage, String fileType
            , int scale) throws IOException{
        bufferedImage= (BufferedImage)bufferedImage
                .getScaledInstance(bufferedImage.getWidth()
                , bufferedImage.getHeight(), scale);
        ImageIO.write(bufferedImage, fileType, new File(pngOutputPath));
    }
}
```

```
package org.deta.tinos.iterator;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import org.json.JSONObject;
import org.json.XML;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
public class IteratorSwap{
    public static String iteratorToJsonString(Gson gson
            , Iterator<Object> iterator){
        return gson.toJson(iterator);
    }
    public static Object stringIteratorToJsonString(Gson gson
            , Iterator<String> iterator) {
        return gson.toJson(iterator);
    }
    public static String iteratorToXml(Gson gson, Iterator<Object> iterator){
        return XML.toString(new JSONObject(gson.toJson(iterator)));
    }
    public static String iteratorToMap(Gson gson, Iterator<Object> iterator){
        return gson.fromJson(new JSONObject(gson.toJson(iterator)).toString()
                , new TypeToken<Map<String, Object>>(){}.getType());
    }
    public static Object[] iteratorToObjectArray(Gson gson
            , Iterator<Object> iterator){
        List<Object> list= new ArrayList<>();
        while(iterator.hasNext()) {
            list.add(iterator.next());
        }
        return list.toArray();
```

```
    }
    public static List<Object> iteratorToList(Gson gson
            , Iterator<Object> iterator){
        List<Object> list= new ArrayList<>();
        while(iterator.hasNext()) {
            list.add(iterator.next());
        }
        return list;
    }
}
```

```
package org.deta.tinos.json;
import java.util.Map;
import java.util.List;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.Hashtable;
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.XML;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
public class JsonSwap{
    public static List<Object> jsonArrayToList(JSONArray jSONArray) {
        List<Object> list= new ArrayList<>();
        for(int i= 0; i< jSONArray.length(); i++){
            Object object= jSONArray.get(i);
            if(object instanceof JSONObject){
                list.add(new Gson().fromJson(jSONArray.getJSONObject(i).toString()
                        , new TypeToken<Map<String, Object>>(){}.getType()));
            }else if(object instanceof String){
                list.add(String.valueOf(object));
            }else if(object instanceof JSONArray){
                list.add(jsonArrayToList(jSONArray.getJSONArray(i)));
            }
        }
        return list;
    }
    public static String jsonObjectToString(JSONObject jSONObject){
        return jSONObject.toString();
    }
    public static Hashtable<String, Object> jsonObjectToHashtable(Gson gson
            ,JSONObject jSONObject){
        return gson.fromJson(jSONObject.toString()
                , (Type) new Hashtable<String, Object>());
    }
    public static String jsonObjectToXml(JSONObject jSONObject){
```

```
        return XML.toString(jSONObject);
    }
}
```

```
package org.deta.tinos.list;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import org.json.JSONObject;
import org.json.XML;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
public class ListSwap{
    public static String listToJsonString(Gson gson, List<Object> list){
        return gson.toJson(list);
    }
    public static Object stringListToJsonString(Gson gson, List<String> list) {
        return gson.toJson(list);
    }
    public static String listToXml(Gson gson, List<Object> list){
        return XML.toString(new JSONObject(gson.toJson(list)));
    }
    public static String listToMap(Gson gson, List<Object> list){
        return gson.fromJson(new JSONObject(gson.toJson(list)).toString()
                , new TypeToken<Map<String, Object>>(){}.getType());
    }
    public static Object[] listToObjectArray(List<Object> list){
        return list.toArray();
    }
    public static Iterator<Object> listToIterator(List<Object> list){
        return list.iterator();
    }
    public static Vector<Object> listToVector(List<Object> list){
        Vector<Object> vector= new Vector<>();
        Iterator<Object> iterator= list.iterator();
        while(iterator.hasNext()) {
            vector.add(iterator.next());
        }
        return vector;
    }
    public static Set<Object> listToSet(List<Object> list){
        Set<Object> sets= new TreeSet<>();
        Iterator<Object> iterator= list.iterator();
        while(iterator.hasNext()) {
```

```
            sets.add(iterator.next());
        }
        return sets;
    }
}
```

```
package org.deta.tinos.list;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.deta.tinos.stable.Stable;
public class ListValidation{
    public static boolean ListSetsCheck(List<Object> list, String setsType) {
        Iterator<Object> iterator= list.iterator();
        while(iterator.hasNext()) {
            Object object= iterator.next();
            if(setsType.equalsIgnoreCase(Stable.STRING_DOUBLE)) {
                if(!(object instanceof Double)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_INT)) {
                if(!(object instanceof Integer)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_FLOAT)) {
                if(!(object instanceof Float)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_STRING)) {
                if(!(object instanceof String)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_SHORT)) {
                if(!(object instanceof Short)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_BOOLEAN)) {
                if(!(object instanceof Boolean)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_LONG)) {
```

```
                if(!(object instanceof Long)) {
                    return false;
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_BYTE)) {
                if(!(object instanceof Byte)) {
                    return false;
                }
            }
        }
        return true;
    }
    public static List<Object> ListSetsFix(List<Object> list
            , String setsType) {
        List<Object> output= new ArrayList<>();
        Iterator<Object> iterator= list.iterator();
        while(iterator.hasNext()) {
            Object object= iterator.next();
            if(setsType.equalsIgnoreCase(Stable.STRING_DOUBLE)) {
                if(!(object instanceof Double)) {
                    output.add((double)0.00);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_INT)) {
                if(!(object instanceof Integer)) {
                    output.add((int)0);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_FLOAT)) {
                if(!(object instanceof Float)) {
                    output.add((float)0.0);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_STRING)) {
                if(!(object instanceof String)) {
                    output.add(Stable.STRING_EMPTY);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_SHORT)) {
```

```
                if(!(object instanceof Short)) {
                    output.add((short)0);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_BOOLEAN)) {
                if(!(object instanceof Boolean)) {
                    output.add(false);
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_LONG)) {
                if(!(object instanceof Long)) {
                    output.add((long)(0));
                }else {
                    output.add(object);
                }
            }
            if(setsType.equalsIgnoreCase(Stable.STRING_BYTE)) {
                if(!(object instanceof Byte)) {
                    output.add((byte)0);
                }else {
                    output.add(object);
                }
            }
        }
        return output;
    }
}
```

```
package org.deta.tinos.map;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
public class MapSwap{
    public static List<Map<String, Object>> mapToList(Map<String, Object> map){
        List<Map<String, Object>> list= new ArrayList<>();
        Iterator<String> iterator= map.keySet().iterator();
        while(iterator.hasNext()) {
            String string= iterator.next();
            Map<String, Object> singer= new HashMap<>();
            singer.put(string, map.get(string));
```

```
            list.add(singer);
        }
        return list;
    }
    public static Hashtable<String, Object> mapToHash(Map<String, Object> map){
        Hashtable<String, Object> hashtable= new Hashtable<>();
        Iterator<String> iterator= map.keySet().iterator();
        while(iterator.hasNext()) {
            String string= iterator.next();
            hashtable.put(string, map.get(string));
        }
        return hashtable;
    }
}
```

```
package org.deta.tinos.matrix;
public class MatrixSwap{
    public static Object[][] matrixInclineSwap(Object[][] objects){
        Object[][] output= new Object[objects[0].length][objects.length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                output[j][i]= objects[i][j];
            }
        }
        return output;
    }
    public static Object[][] matrixPostSwap(Object[][] objects){
        Object[][] output= new Object[objects.length][objects[0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                output[i][objects[0].length- j]= objects[i][j];
            }
        }
        return output;
    }
    public static Object[][] matrixInSwap(Object[][] objects){
        Object[][] output= new Object[objects.length][objects[0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                output[objects.length- i][j]= objects[i][j];
            }
        }
        return output;
    }
    public static Object[][] matrixReverseSwap(Object[][] objects){
        Object[][] output= new Object[objects.length][objects[0].length];
        for(int i= 0; i< objects.length; i++) {
```

```
            for(int j= 0; j< objects[0].length; j++) {
                output[objects.length- i][objects[0].length- j]= objects[i][j];
            }
        }
        return output;
    }
    public static Object[] matrixToArray(Object[][] objects){
        Object[] output= new Object[objects.length* objects[0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                output[i* objects.length+ j]= objects[i][j];
            }
        }
        return output;
    }
}
```

```
package org.deta.tinos.matrix;
import org.deta.tinos.stable.Stable;
public class MatrixValidation{
    public static boolean matrixCheck(Object[][] objects
            , String attribute){
        if(attribute.equalsIgnoreCase(Stable.STRING_BOOLEAN)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Boolean)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_INT)) {
            for(int i= 0; i<objects.length; i++) {
                for(int j= 0; j<objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Integer)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_LONG)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Long)) {
                        return false;
                    }
                }
```

```
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_DOUBLE)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Double)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_FLOAT)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Float)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_STRING)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof String)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_SHORT)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Short)) {
                        return false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_BYTE)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Byte)) {
                        return false;
                    }
                }
            }
        }
```

```
        return true;
    }
    public static Object[][] matrixFix(Object[][] objects
            , String attribute){
        if(attribute.equalsIgnoreCase(Stable.STRING_BOOLEAN)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Boolean)) {
                        objects[i][j]= false;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_INT)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Integer)) {
                        objects[i][j]= (int)0;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_LONG)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Long)) {
                        objects[i][j]= (long)0;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_DOUBLE)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Double)) {
                        objects[i][j]= (double)0.0;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_FLOAT)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Float)) {
                        objects[i][j]= (float)0.0;
                    }
                }
```

```
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_STRING)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof String)) {
                        objects[i][j]= "";
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_SHORT)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Short)) {
                        objects[i][j]= (short)0;
                    }
                }
            }
        }
        if(attribute.equalsIgnoreCase(Stable.STRING_BYTE)) {
            for(int i= 0; i< objects.length; i++) {
                for(int j= 0; j< objects[0].length; j++) {
                    if(!(objects[i][j] instanceof Byte)) {
                        objects[i][j]= (byte)0;
                    }
                }
            }
        }
        return objects;
    }
}
```

```
package org.deta.tinos.matrix3D;
public class Matrix3DSwap{
    public static Object[][][] matrixShiftSwapXYZ(Object[][][] objects){
        Object[][][] output
        = new Object[objects[0][0].length][objects.length][objects[0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[k][i][j]= objects[i][j][k];
                }
            }
        }
        return output;
    }
```

```
    public static Object[][][] matrixInclineSwapXY(Object[][][] objects){
        Object[][][] output
        = new Object[objects[0].length][objects.length][objects[0][0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[j][i][k]= objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[][][] matrixInclineSwapXZ(Object[][][] objects){
        Object[][][] output
        = new Object[objects[0][0].length][objects[0].length][objects.length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[k][j][i]= objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[][][] matrixInclineSwapYZ(Object[][][] objects){
        Object[][][] output
        = new Object[objects.length][objects[0][0].length][objects[0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[i][k][j]= objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[][][] matrixReverseSwap(Object[][][] objects){
        Object[][][] output= new Object[objects.length][objects[0].length][];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[objects.length- i][objects[0].length- j]
                            [objects[0][0].length- k]= objects[i][j][k];
                }
            }
        }
        return output;
```

```
    }
    public static Object[][][] matrixReverseSwapXY(Object[][][] objects){
        Object[][][] output= new Object[objects.length][objects[0].length][];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[objects.length- i][objects[0].length- j][k]
                            = objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[][][] matrixReverseSwapXZ(Object[][][] objects){
        Object[][][] output= new Object[objects.length][objects[0].length][];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[objects.length- i][j][objects[0][0].length- k]
                            = objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[][][] matrixReverseSwapYZ(Object[][][] objects){
        Object[][][] output= new Object[objects.length][objects[0].length][];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[i][objects[0].length- j][objects[0][0].length- k]
                            = objects[i][j][k];
                }
            }
        }
        return output;
    }
    public static Object[] matrixToArray(Object[][][] objects){
        Object[] output= new Object[objects.length
                                    * objects[0].length* objects[0][0].length];
        for(int i= 0; i< objects.length; i++) {
            for(int j= 0; j< objects[0].length; j++) {
                for(int k= 0; k< objects[0][0].length; k++) {
                    output[i* objects.length* objects[0].length
                           + j* objects[0].length+ k]= objects[i][j][k];
                }
            }
```

```
        }
        return output;
    }
}
```

```
package org.deta.tinos.object;
import com.google.gson.Gson;
public class ObjectSwap{
    public static String objectToJsonString(Gson gson, Object object){
        return gson.toJson(object);
    }
}
```

```
package org.deta.tinos.stable;
public interface Stable {
    public static final String DOUBLE_INDICATION_NULL= "NULL";
    public static final String DOUBLE_INDICATION_TIVE= "TIVE";
    public static final String SUPPRESS_WARNINGS_DEPRECATION
    = "deprecation";
    public static final String SUPPRESS_WARNINGS_STATIC_ACCESS
    = "static-access";
    public static final String INVALID_DOUBLE_FORMAT
    = "Invalid Double Format!";
    public static final String STRING_EMPTY= "";
    public static final String STRING_PER= ":";
    public static final String STRING_INT= "int";
    public static final String STRING_DOUBLE= "double";
    public static final String STRING_STRING= "string";
    public static final String STRING_FLOAT= "float";
    public static final String STRING_BYTE= "byte";
    public static final String STRING_BOOLEAN= "boolean";
    public static final String STRING_LONG= "long";
    public static final String STRING_SHORT= "short";
    public static final String STRING_BJTIME= "http://www.bjtime.cn";
    public static final String CHARSET_GBK= "GBK";
    public static final String CHARSET= "charset";
    public static final String TYPE= "type";
    public static final String CHARSET_UTF8= "UTF8";
    public static final String CHARSET_UTF_8= "UTF-8";
    public static final String CHARSET_GB2312= "GB2312";
    public static final String CHARSET_ASCII= "ASCII";
    public static final String CHARSET_UNICODE= "UNICODE";
    public static final String CHARSET_ISO_8859_1= "ISO-8859-1";
    public static final String CONTENT_TYPE= "Content-Type";
    public static final String APPLICATION_XML= "application/xml";
    public static final String APPLICATION_JSON= "application/json";
    public static final String APPLICATION_JSON_UTF8
```

```
    = "application/json;charset=UTF-8";
    public static final String HTTP_POST= "POST";
    public static final String HTTP_GET= "GET";
}
```

```
package org.deta.tinos.stock;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class StockCode{
    public List<String> list;
    public List<String> todayList;
    public void readDBcodeTOList(String code) throws IOException {
        list= new ArrayList<String>();
        File afile
        = new File("StockString.OTOO_STOCK_DB_OLD+code+StockString.OTOO_TXT");
        BufferedReader areader= null;
        String atempString= "StockString.OTOO_EMPTY";
        areader= new BufferedReader(new FileReader(afile));
        while ((atempString= areader.readLine())!= null) {
            if(atempString!= null)
                if(!atempString.equals("StockString.OTOO_EMPTY")){
                    list.add(atempString);
                }
        }
        areader.close();
    }
    public void readTodaycodeTOList() throws IOException {
        todayList= new ArrayList<String>();
        File afile= new File("tockString.STOCK_DB_TODAY");
        BufferedReader areader= null;
        String atempString= "StockString.OTOO_EMPTY";
        areader= new BufferedReader(new FileReader(afile));
        while ((atempString= areader.readLine())!= null) {
            if(atempString!= null)
                if(!atempString.equals("StockString.OTOO_EMPTY")){
                    todayList.add(atempString);
                }
        }
        areader.close();
    }
    public String readStringTOFormatWithoutCode(String input) throws IOException {
```

```
        String output= null;
        Pattern p= Pattern.compile("\"(.*?)\"");
        Matcher m= p.matcher(input);
        if(m.find())
            if(!m.group(0).equals("\"\"")) {
                output= m.group(0).replace("\"", "");
            }
        return output;
    }
    public String readStringTOFormatWithCode(String input) throws IOException {
        String output= null;
        Pattern p0= Pattern.compile("r_(.*?)=");
        Matcher m0= p0.matcher(input);
        Pattern p= Pattern.compile("\"(.*?)\"");
        Matcher m= p.matcher(input);
        if(m.find())
            if(!m.group(0).equals("\"\"")) {
                if(m0.find()) {
                    output= m0.group(0).replace("r_", "").replace("=", "");
                }
                output+= ","+ m.group(0).replace("\"", "");
            }
        return output;
    }
    public static void main(String [] argv)
            throws IOException, InterruptedException {
        //0to0 a=new 0to0();
    }
}
```

```
package org.deta.tinos.string;
import java.util.HashMap;
import java.util.Map;
public class QuickLuoyaoguang4D{
    Map<String, Boolean> find= new HashMap<>();
    public void quick4DStringArray(String[] a, int lp, int rp, int scale) {
        quick4DString(a, lp, rp, 0);
        for(int k= 1; k< scale; k++) {
            quick4DString(a, lp, rp, k);
        }
    }
    public void quick4DString(String[] a, int lp, int rp, int scale) {
        if(lp< rp){
            int c= rp- lp; if(c< 7){
                int j;
                for(int i= 1+ lp; i<= lp+ c; i++){
                    j= i;
```

```
                while(j>= 1+ lp){
                    if(a[j].length()> scale&& a[j- 1].length()> scale) {
                        if(a[j].toLowerCase().charAt(scale)
                                < a[j- 1].toLowerCase().charAt(scale)){
                            conditionSwap(a, scale, j);
                        }else if(a[j].toLowerCase().charAt(scale)
                                == a[j-1].toLowerCase().charAt(scale)) {
                            if(a[j].charAt(scale)< a[j- 1].charAt(scale)) {
                                conditionSwap(a, scale, j);
                            }
                        }
                    }
                    j--;
                }
            }
            return;
        }
        int pos= partitionString(a, lp, rp, scale);
        quick4DString(a, lp, pos- 1, scale);
        quick4DString(a, pos+ 1, rp, scale);
    }
}
private void conditionSwap(String[] a, int scale, int j) {
    boolean find= true;
    for(int p= 0; p< scale; p++) {
        if(a[j].charAt(p)!= a[j- 1].charAt(p)) {
            find= false;
        }
    }
    if(find) {
        String temp= a[j];
        a[j]= a[j-1];
        a[j-1]= temp;
    }
}
private int partitionString(String[] a, int lp, int rp, int scale) {
    String x= a[lp];
    if(!(a[lp].length()<= scale|| a[rp].length()<= scale)){
        x= a[lp].toLowerCase().charAt(scale)
                < a[rp].toLowerCase().charAt(scale)? a[lp]: a[rp];
    }
    int lp1= lp;
    while(lp1< rp){
        while(innerConditionUp(a, scale, x, lp1, rp)){
            lp1++;
        }
        while(innerConditionDown(a, scale, x, rp)){
```

```
                rp--;
            }
            if(lp1< rp){
                boolean find= true;
                for(int p= 0; p< scale; p++) {
                    if(a[rp].charAt(p)!= a[lp1].charAt(p)) {
                        find= false;
                    }
                }
                if(find) {
                    String temp= a[rp];
                    a[rp]= a[lp1];
                    a[lp1]= temp;
                }else {
                    lp1++;
                }
            }
        }
        if(lp1< rp) {
            a[lp]= a[rp]; a[rp]= x;
        }
        return rp;
    }
    private boolean innerConditionUp(String[] a, int scale
            , String x, int lp1, int rp) {
        if(lp1>= a.length) {
            return false;
        }
        if(a[lp1].length()<= scale|| x.length()<= scale) {
            return true;
        }
        if(!(a[lp1].toLowerCase().charAt(scale)
                > x.toLowerCase().charAt(scale)|| lp1>= rp)) {
            return true;
        }
        return false;
    }
    private boolean innerConditionDown(String[] a, int scale, String x, int rp) {
        if(rp>= a.length){
            return false;
        }
        if(rp< 0) {
            return false;
        }
        if(a[rp].length()<= scale|| x.length()<= scale) {
            return true;
        }
```

```
        if(a[rp].toLowerCase().charAt(scale)
                > x.toLowerCase().charAt(scale)) {
            return true;
        }
        return false;
    }
}
```

```
package org.deta.tinos.string;
public class StringSequency{
    public static String stringToSequency(String string){
        char[] chars= string.toCharArray();
        for(int i= 0; i< chars.length; i++) {
            for(int j= 0; j< chars.length; j++) {
                if(chars[i]< chars[j]) {
                    char temp= chars[i];
                    chars[i]= chars[j];
                    chars[j]= temp;
                }
            }
        }
        return String.valueOf(chars);
    }
    public static String[] stringArrayToSequencyArrayIncrement(String[] string
            , int scale){
        //比较头
        for(int i= 0; i< string.length; i++) {
            for(int j= 0; j< string.length; j++) {
                //区别字母
                if(string[i].toLowerCase().charAt(0)
                        < string[j].toLowerCase().charAt(0)) {
                    //区别大小写
                    String temp= string[i].toString();
                    string[i]= string[j].toString();
                    string[j]= temp.toString();
                }
            }
        }
        //比较身
        for(int k= 1; k< scale; k++) {
            for(int i= 0; i< string.length; i++) {
                for(int j= 0; j< string.length; j++) {
                    if(string[i].length()> k&& string[j].length()> k) {
                        if(string[i].toLowerCase().charAt(k)
                                < string[j].toLowerCase().charAt(k)) {
                            boolean find= true;
                            for(int p= 0; p< k; p++) {
```

```
                            if(string[i].charAt(p)!= string[j].charAt(p)) {
                                find= false;
                            }
                        }
                        if(find) {
                            String temp= string[i].toString();
                            string[i]= string[j].toString();
                            string[j]= temp.toString();
                        }
                    }
                }
            }
        }
        return string;
    }
    public static String[] stringArrayToSequencyArrayDecrement(String[] string
            , int scale){
        //比较头
        for(int i= 0; i< string.length; i++) {
            for(int j= 0; j< string.length; j++) {
                //区别字母
                if(string[i].toLowerCase().charAt(0)
                        < string[j].toLowerCase().charAt(0)) {
                    //区别大小写
                    String temp= string[i].toString();
                    string[i]= string[j].toString();
                    string[j]= temp.toString();
                }else if(string[i].toLowerCase().charAt(0)
                        == string[j].toLowerCase().charAt(0)) {
                    //区别大小写
                    if(string[i].charAt(0)<string[j].charAt(0)) {
                        String temp= string[i].toString();
                        string[i]= string[j].toString();
                        string[j]= temp.toString();
                    }
                }
            }
        }
        //比较身
        for(int k= 1; k< scale; k++) {
            for(int i= 0; i< string.length; i++) {
                for(int j= 0; j< string.length; j++) {
                    if(string[i].length()> k&& string[j].length()> k) {
                        if(string[i].toLowerCase().charAt(k)
                                < string[j].toLowerCase().charAt(k)) {
                            boolean find= true;
```

```
                            for(int p= 0; p< k; p++) {
                                if(string[i].charAt(p)!= string[j].charAt(p)) {
                                    find= false;
                                }
                            }
                            if(find) {
                                String temp= string[i].toString();
                                string[i]= string[j].toString();
                                string[j]= temp.toString();
                            }
                        }else if(string[i].toLowerCase().charAt(k)
                                == string[j].toLowerCase().charAt(k)) {
                            //区别大小写
                            if(string[i].charAt(k)<string[j].charAt(k)) {
                                boolean find= true;
                                for(int p= 0; p< k; p++) {
                                    if(string[i].charAt(p)!= string[j].charAt(p)) {
                                        find= false;
                                    }
                                }
                                if(find) {
                                    String temp= string[i].toString();
                                    string[i]= string[j].toString();
                                    string[j]= temp.toString();
                                }
                            }
                        }
                    }
                }
            }
        }
        return string;
    }
}
```

```
package org.deta.tinos.string;
import org.json.XML;
import java.util.List;
import java.util.Map;
import org.json.JSONArray;
import org.json.JSONObject;
import java.io.UnsupportedEncodingException;
import java.math.BigDecimal;
import java.util.ArrayList;
import com.google.gson.Gson;
import org.deta.tinos.json.JsonSwap;
import org.deta.tinos.stable.Stable;
```

```
import com.google.gson.reflect.TypeToken;
@SuppressWarnings(Stable.SUPPRESS_WARNINGS_DEPRECATION)
public class StringSwap{
    public static String stringToJsonObject(Gson gson, String string){
        return gson.toJson(string);
    }
    public static String stringToXml(Gson gson, String string){
        JSONObject jSONObject= new JSONObject(gson.toJson(string));
        return XML.toString(jSONObject);
    }
    public static String[] stringToArray(String stopBy, String string){
        String[] strings= string.split(stopBy);
        return strings;
    }
    public static List<Object> stringToList(Gson gson, String string
            , String key){
        JSONArray jSONArray= XML.toJSONObject(string).getJSONArray(key);
        List<Object> list= new ArrayList<>();
        for(int i= 0; i< jSONArray.length(); i++){
            Object object= jSONArray.get(i);
            if(object instanceof JSONObject){
                list.add(new Gson().fromJson(jSONArray
                        .getJSONObject(i).toString(), new TypeToken<Map<String
                        , Object>>(){}.getType()));
            }else if(object instanceof String){
                list.add(String.valueOf(object));
            }else if(object instanceof JSONArray){
                list.add(JsonSwap.jsonArrayToList(jSONArray.getJSONArray(i)));
            }
        }
        return list;
    }
    public static Map<String, Object> stringToMap(Gson gson, String string){
        JSONObject jSONObject= new JSONObject(gson.toJson(string));
        return gson.fromJson(jSONObject.toString()
                , new TypeToken<Map<String, Object>>(){}.getType());
    }
    public static String stringToURIencode(String string){
        return java.net.URLEncoder.encode(string);
    }
    public static String uRIencodeToURIdecode(String string){
        return java.net.URLDecoder.decode(string);
    }
    public static int[] stringToCharASCII(String string){
        int[] charASCII= new int[string.length()];
        for(int i= 0; i< string.length(); i++){
            charASCII[i]= string.charAt(i);
```

```
        }
        return charASCII;
    }
    public static String charsetSwap(String string,String inputCharset
            , String outputCharset)
                    throws UnsupportedEncodingException{
        String output= new String(string.getBytes(inputCharset), outputCharset);
        return output;
    }
    public static double stringDoubleToDouble(String stringDouble)
            throws Exception {
        if(stringDouble.toUpperCase().contains(Stable.DOUBLE_INDICATION_NULL)
                || stringDouble.toUpperCase().contains(Stable
                        .DOUBLE_INDICATION_TIVE)) {
            throw new Exception(Stable.INVALID_DOUBLE_FORMAT);
        }
        double output= Double.valueOf(stringDouble);
        return output;
    }
    public static String stringDoubleToBigDecimalRemainder(String stringDouble
            , int newScale) throws Exception {
        if(stringDouble.toUpperCase().contains(Stable.DOUBLE_INDICATION_NULL)
                || stringDouble.toUpperCase().contains(Stable
                        .DOUBLE_INDICATION_TIVE)) {
            throw new Exception(Stable.INVALID_DOUBLE_FORMAT);
        }
        BigDecimal output= new BigDecimal(Double.valueOf(stringDouble))
                .setScale(newScale, BigDecimal.ROUND_HALF_UP);
        return output.toString();
    }
}
```

```
package org.deta.tinos.string;
import java.util.Map;
import org.deta.tinos.stable.Stable;
public class StringValidation{
    public static String stringCodeFilter(String string, String swapSymbol
            , String collection){
        StringBuilder stringBuilder= new StringBuilder();
        for(int i= 0; i< string.length(); i++) {
            if(collection.contains(Stable.STRING_EMPTY+ string.charAt(i))){
                stringBuilder.append(swapSymbol);
            }else {
                stringBuilder.append(string.charAt(i));
            }
        }
        return stringBuilder.toString();
```

```
    }
    public static String stringCodeFilter(String string, String swapSymbol
            , Map<String, Boolean> collection){
        StringBuilder stringBuilder= new StringBuilder();
        for(int i= 0; i< string.length(); i++) {
            if(collection.containsKey(Stable.STRING_EMPTY+ string.charAt(i))){
                stringBuilder.append(swapSymbol);
            }else {
                stringBuilder.append(string.charAt(i));
            }
        }
        return stringBuilder.toString();
    }
}
```

```
package org.deta.tinos.stringBuilder;
import java.util.Map;
import org.json.JSONObject;
import org.json.XML;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
public class StringBuilderSwap{
    public static char[] stringBuilderToCharArray(StringBuilder stringBuilder) {
        return stringBuilder.toString().toCharArray();
    }
    public static String stringBuilderToString(StringBuilder stringBuilder) {
        return stringBuilder.toString();
    }
    public static String stringBuilderToJson(Gson gson
            , StringBuilder stringBuilder) {
        return gson.toJson(stringBuilder.toString());
    }
    public static String stringBuilderToXml(Gson gson
            , StringBuilder stringBuilder){
        JSONObject jSONObject
        = new JSONObject(gson.toJson(stringBuilder.toString()));
        return XML.toString(jSONObject);
    }
    public static String[] stringBuilderToSplitArray(String stopBy
            , StringBuilder stringBuilder){
        String[] strings= stringBuilder.toString().split(stopBy);
        return strings;
    }
    public static Map<String, Object> stringBuilderToMap(Gson gson
            , StringBuilder stringBuilder){
        JSONObject jSONObject
        = new JSONObject(gson.toJson(stringBuilder.toString()));
```

```
        return gson.fromJson(jSONObject.toString()
                , new TypeToken<Map<String, Object>>(){}.getType());
    }
}
```

```
package org.deta.tinos.tree;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.Vector;
import org.json.JSONObject;
import org.json.XML;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
public class TreeSwap{
    public static String treeToJsonString(Gson gson, Set<Object> sets){
        return gson.toJson(sets);
    }
    public static Object stringTreeToJsonString(Gson gson, Set<String> sets) {
        return gson.toJson(sets);
    }
    public static String treeToXml(Gson gson, Set<Object> sets){
        return XML.toString(new JSONObject(gson.toJson(sets)));
    }
    public static String treeToMap(Gson gson, Set<Object> sets){
        return gson.fromJson(new JSONObject(gson.toJson(sets)).toString()
                , new TypeToken<Map<String, Object>>(){}.getType());
    }
    public static Object[] treeToObjectArray(Set<Object> sets){
        return sets.toArray();
    }
    public static Iterator<Object> treeToIterator(Set<Object> sets){
        return sets.iterator();
    }
    public static Vector<Object> treeToVector(Set<Object> sets){
        Vector<Object> vector= new Vector<>();
        Iterator<Object> iterator= sets.iterator();
        while(iterator.hasNext()) {
            vector.add(iterator.next());
        }
        return vector;
    }
}
```

```
package org.deta.tinos.tsp;
import java.util.ArrayList;
import java.util.List;
```

```
public class TSP{
    public static int small=1000000000;
    public static void main(String []argv){
        int []x= new int[11];
        int []y= new int[11];
        x[0]= 0;y[0]= 0;
        x[1]= 1;y[1]= 0;
        x[2]= 2;y[2]= 10;
        x[3]= 12;y[3]= 16;
        x[4]= 6;y[4]= 6;
        x[5]= 8;y[5]= 22;
        x[6]= 16;y[6]= 30;
        x[7]= 22;y[7]= 18;
        x[8]= 18;y[8]= 20;
        x[9]= 10;y[9]= 10;
        x[10]= 1;y[10]= 1;
        node first= new node();
        first.x=0;
        first.y=0;
        first.used=0;
        List<node>nodes= new ArrayList<node>();
        findpath(first,x,y,nodes);
        System.out.print(1);
        sort(nodes,first.x,first.y,x.length);
        System.out.print(2);
    }
    public static void sort( List<node> nodes,int x,  int y,int N){
        node temp=nodes.get(nodes.size()-1);
        node temp1=nodes.get(nodes.size()-1);
        System.out.println("the short one");
        node []out=new node[N];
        while(temp!=null){
            out[temp.used]=temp;
            temp=temp.prev;
        }
        while(temp1!=null){
            out[temp1.used]=temp1;
            temp1=temp1.next;
        }
        for(int i1=0;i1<out.length;i1++){
            if(out[i1]!=null){
                System.out.println("x:"+out[i1].x+"y:"+out[i1].y);
            }
        }
        System.out.println("the shortest distance:"+out[out.length-1].total_dis);
    }
    public static int find(node first,int i,int[] x,int[] y){
```

```
        if(first.x== x[i] && first.y== y[i]){
            return 1;
        }
        node temp= first;
        node temp1= first;
        while(temp.prev!= null){
            temp= temp.prev;
            if(temp.x== x[i]&& temp.y== y[i]){
                return 1;
            }
        }
        while(temp1.next!=null){
            temp1= temp1.next;
            if (temp1.x== x[i] && temp1.y== y[i]) {
                return 1;
            }
        }
        return 0;
    }
    public static int findSmall(node first,int []x,int []y){
        while(first.next!=null) {
            first= first.next;
        }
        if(first.x==x[x.length-1]&& first.y==y[y.length-1]){
            if(first.used==x.length-1){
                if(first.total_dis<small){
                    small=first.total_dis;
                    return 1;
                }
                //System.out.println(first.u
            }
        }
        return 0;
    }
    public static void findpath(node first, int[]x, int[]y, List<node> nodes){
        for(int i=0;i<x.length;i++){
            node first_copy= new node();
            first_copy.x= first.x;
            first_copy.y= first.y;
            first_copy.used=first.used;
            first_copy.total_dis= first.total_dis;
            first_copy.next=first.next;
            first_copy.prev=first.prev;
            int find= find(first_copy,i,x,y);
            // int findSmall= findSmall(first_copy,i,x,y);
            if(find== 0){
                int findSmall=findSmall(first_copy,x,y);
```

```
                //if(temp.used>=x.length-2)
                //{
                //}
                node next= new node();
                next.x= x[i];
                next.y= y[i];
                int dis= Math.abs(x[i] - first.x)
                        + Math.abs(y[i] - first.y);
                next.total_dis= first_copy.total_dis + dis;
                next.used= first_copy.used+1;
                first_copy.next= next;
                next.prev= first_copy;
                first_copy= first_copy.next;
                findSmall= findSmall(first_copy,x,y);
                if(findSmall== 1){
                    // small= temp.total_dis;
                    // nodes.clear();
                    nodes.add(first_copy);
                }
                findpath(first_copy, x, y,nodes);
            }//
        }
    }
}
class node{
    public int used= 0;
    public int x;
    public int y;
    public int total_dis= 0;
    public node prev;
    public node next;
    public int index;
}
```

```
package org.deta.tinos.tsp;
import java.util.ArrayList;
import java.util.List;
public class TSPEuler{
    public static int small= 1000000000;
    public static int te= 0;
    public static void main(String[] argv){
        int []x= new int[41];
        int []y= new int[41];
        x[0]= 0;y[0]= 0;
        x[1]= 1;y[1]= 20;
        x[2]= 12;y[2]= 2;
        x[3]= 3;y[3]= 23;
```

```
        x[4]= 4;y[4]= 4;
        x[5]= 15;y[5]= 25;
        x[6]= 6;y[6]= 6;
        x[7]= 17;y[7]= 7;
        x[8]= 8;y[8]= 28;
        x[9]= 19;y[9]= 1;
        x[10]= 10;y[10]= 10;
        x[11]= 11;y[11]= 11;
        x[12]= 12;y[12]= 12;
        x[13]= 13;y[13]= 13;
        x[14]= 14;y[14]= 14;
        x[15]= 15;y[15]= 15;
        x[16]= 16;y[16]= 16;
        x[17]= 17;y[17]= 17;
        x[18]= 18;y[18]= 18;
        x[19]= 19;y[19]= 19;
        x[20]= 20;y[20]= 20;
        x[21]= 21;y[21]= 21;
        x[22]= 22;y[22]= 22;
        x[23]= 23;y[23]= 23;
        x[24]= 24;y[24]= 24;
        x[25]= 25;y[25]= 25;
        x[26]= 26;y[26]= 26;
        x[27]= 27;y[27]= 27;
        x[28]= 28;y[28]= 28;
        x[29]= 29;y[29]= 29;
        x[30]= 30;y[30]= 30;
        x[31]= 31;y[31]= 31;
        x[32]= 32;y[32]= 32;
        x[33]= 33;y[33]= 33;
        x[34]= 34;y[34]= 34;
        x[35]= 35;y[35]= 35;
        x[36]= 36;y[36]= 36;
        x[37]= 37;y[37]= 37;
        x[38]= 38;y[38]= 38;
        x[39]= 39;y[39]= 39;
        x[40]= 1;y[40]= 1;
        /*
       x[41]= 1;y[41]= 1;
/*
        x[42]= 1;y[42]= 1;
     /*
       x[43]= 43;y[43]= 33;
       x[44]= 1;y[44]= 1;
      /*
       x[45]= 45;y[45]= 35;
       x[46]= 46;y[46]= 36;
```

```
        x[47]= 47;y[47]= 37;
        x[48]= 48;y[48]= 38;
        x[49]= 49;y[49]= 39;
        x[50]= 1;y[50]= 1;
          */
         node first= new node();
         first.x=0;
         first.y=0;
         first.used=0;
         List<node>nodes= new ArrayList<node>();
         if(x.length>13){
             findpath(first,x,y,nodes);
         }else{
             findrandom(first,x,y,nodes);
         }
         System.out.print(1);
         sort(nodes,first.x,first.y,x.length);
         System.out.print(2);
    }
    public static void findrandom(node first, int[]x
            , int[]y, List<node> nodes){
         int min=999999999;
         int big=0;
         for(int i=0;i<x.length;i++){
             node first_copy= new node();
             first_copy.x= first.x;
             first_copy.y= first.y;
             first_copy.used= first.used;
             first_copy.total_dis= first.total_dis;
             first_copy.next= first.next;
             first_copy.prev= first.prev;
             int find= find(first_copy,i,x,y);
             if(find== 0){
                 te++;
                 node next= new node();
                 next.x= x[i];
                 next.y= y[i];
                 int dis= Math.abs(x[i]- first.x)
                         + Math.abs(y[i]- first.y);
                 next.total_dis= first_copy.total_dis+ dis;
                 next.used= first_copy.used+ 1;
                 first_copy.next= next;
                 next.prev= first_copy;
                 first_copy= first_copy.next;
                 int findSmall= findSmall(first_copy, x, y);
                 if(findSmall== 1){
                     nodes.add(first_copy);
```

```
                }
                int cc= 0;
                if(first_copy.total_dis>big){
                    cc+= 1;
                    big= first_copy.total_dis;
                }else if(first_copy.total_dis< min){
                    cc+= 1;
                    min= first_copy.total_dis;
                }
                if(cc>0) {
                    findrandom(first_copy, x, y, nodes);
                }
            }
        }
    }
    public static void sort( List<node> nodes,int x, int y, int N){
        node temp= nodes.get(nodes.size()- 1);
        node temp1= nodes.get(nodes.size()- 1);
        System.out.println("the short one");
        node []out= new node[N];
        while(temp!= null){
            out[temp.used]= temp;
            temp= temp.prev;
        }
        while(temp1!= null){
            out[temp1.used]= temp1;
            temp1= temp1.next;
        }
        for(int i1= 0;i1< out.length; i1++){
            if(out[i1]!= null){
                System.out.println("x:"+ out[i1].x
                        + "y:"+ out[i1].y);
            }
        }
        System.out.println("the shortest distance:"
                + out[out.length- 1].total_dis);
    }
    public static int find(node first, int i, int[] x, int[] y){
        if(first.x== x[i]&& first.y== y[i]){
            return 1;
        }
        node temp= first;
        node temp1= first;
        while(temp.prev!= null){
            temp= temp.prev;
            if(temp.x== x[i]&& temp.y== y[i]){
                return 1;
```

```
            }
            if(temp.x== x[x.length- 1]&& temp.y== y[y.length- 1]){
                if(temp.used!= x.length){
                    return 1;
                }
            }
            if(temp.x== x[0]&& temp.y== y[0]){
                if(temp.used!= 0){
                    return 1;
                }
            }
        }
        while(temp1.next!= null){
            temp1= temp1.next;
            if (temp1.x== x[i]&& temp1.y== y[i]) {
                return 1;
            }
            if(temp1.x== x[x.length- 1]
                    && temp1.y== y[y.length- 1]){
                if(temp1.used!= x.length){
                    return 1;
                }
            }
            if(temp1.x== x[0]&& temp1.y== y[0]){
                if(temp1.used!= 0){
                    return 1;
                }
            }
        }
        if(temp.used!= 0){
            return 1;
        }
        if(temp.x!= x[0]|| temp.y!= y[0]){
            return 1;
        }
        return 0;
    }
    public static int findSmall(node first,int []x,int []y){
        while(first.next!= null) {
            first= first.next;
        }
        if(first.x== x[x.length- 1]&& first.y== y[y.length- 1]){
            if(first.used== x.length- 1){
                if(first.total_dis< small){
                    small= first.total_dis;
                    return 1;
                }
```

```
            }
        }
        return 0;
    }
    public static void findpath(node first, int[]x
            , int[]y, List<node> nodes){
        int min= 999999999;
        int big= 0;
        for(int i= 0; i< x.length; i++){
            int find= find(first,i,x,y);
            if(find== 0) {
                int dis= Math.abs(x[i]- first.x)
                        + Math.abs(y[i]- first.y);
                int total_dis= first.total_dis+ dis;
                if (total_dis>= big) {
                    big= total_dis;
                }
                if (total_dis<= min) {
                    min= total_dis;
                }
            }
        }
        for(int i= 0; i< x.length; i++){
            node first_copy= new node();
            first_copy.x= first.x;
            first_copy.y= first.y;
            first_copy.used= first.used;
            first_copy.total_dis= first.total_dis;
            first_copy.next= first.next;
            first_copy.prev= first.prev;
            int find= find(first_copy,i,x,y);
            if(find== 0){
                te++;
                node next= new node();
                next.x= x[i];
                next.y= y[i];
                int dis= Math.abs(x[i]- first.x)
                        + Math.abs(y[i]- first.y);
                next.total_dis= first_copy.total_dis+ dis;
                next.used= first_copy.used+ 1;
                first_copy.next= next;
                next.prev= first_copy;
                first_copy= first_copy.next;
                int findSmall= findSmall(first_copy,x,y);
                if(findSmall== 1){
                    nodes.add(first_copy);
                }
```

```
                int cc=0;
                if(first_copy.total_dis>= big*1){
                    cc+= 1;
                    //big=first_copy.total_dis;
                }else
                    if(first_copy.total_dis <= min*1){
                        cc+= 1;
                        //min=first_copy.total_dis;
                    }
                if(cc>0) {
                    findpath(first_copy, x, y,nodes);
                }
            }//
        }
    }
}
```

```
package org.deta.tinos.tsp;
import java.util.ArrayList;
import java.util.List;
public class YaoguangEulerTSP{
    public static int small= 1000000000;
    public static int te= 0;
    public static void main(String []argv){
        int []x= new int[43];
        int []y= new int[43];
        x[0]= 0;y[0]= 0;
        x[1]= 1;y[1]= 20;
        x[2]= 12;y[2]= 2;
        x[3]= 3;y[3]= 23;
        x[4]= 4;y[4]= 4;
        x[5]= 15;y[5]= 25;
        x[6]= 6;y[6]= 6;
        x[7]= 17;y[7]= 7;
        x[8]= 8;y[8]= 28;
        x[9]= 19;y[9]= 1;
        x[10]= 10;y[10]= 10;
        x[11]= 11;y[11]= 11;
        x[12]= 12;y[12]= 12;
        x[13]= 13;y[13]= 13;
        x[14]= 14;y[14]= 14;
        x[15]= 15;y[15]= 15;
        x[16]= 16;y[16]= 16;
        x[17]= 17;y[17]= 17;
        x[18]= 18;y[18]= 18;
        x[19]= 19;y[19]= 19;
        x[20]= 20;y[20]= 20;
```

```
        x[21]= 21;y[21]= 21;
        x[22]= 22;y[22]= 22;
        x[23]= 23;y[23]= 23;
        x[24]= 24;y[24]= 24;
        x[25]= 25;y[25]= 25;
        x[26]= 26;y[26]= 26;
        x[27]= 27;y[27]= 27;
        x[28]= 28;y[28]= 28;
        x[29]= 29;y[29]= 29;
        x[30]= 30;y[30]= 30;
        x[31]= 31;y[31]= 31;
        x[32]= 32;y[32]= 32;
        x[33]= 33;y[33]= 33;
        x[34]= 34;y[34]= 34;
        x[35]= 35;y[35]= 35;
        x[36]= 36;y[36]= 36;
        x[37]= 37;y[37]= 37;
        x[38]= 38;y[38]= 38;
        x[39]= 39;y[39]= 39;
        x[40]= 31;y[40]= 21;
        x[41]= 11;y[41]= 21;
        x[42]= 1;y[42]= 1;
        /*
       x[41]= 1;y[41]= 1;
/*
        x[42]= 1;y[42]= 1;
     /*
        x[43]= 43;y[43]= 33;
        x[44]= 1;y[44]= 1;
      /*
        x[45]= 45;y[45]= 35;
        x[46]= 46;y[46]= 36;
        x[47]= 47;y[47]= 37;
        x[48]= 48;y[48]= 38;
        x[49]= 49;y[49]= 39;
        x[50]= 1;y[50]= 1;
          */
        node first= new node();
        first.x= 0;
        first.y= 0;
        first.used= 0;
        List<node>nodes= new ArrayList<node>();
        if(x.length> 13&& x.length<= 41){
            findpath(first, x, y, nodes);
        }else if(x.length<= 13){
            findrandom(first, x, y, nodes);
        }else{
```

```
            findstate(first, x, y, nodes);
        }
        System.out.print(1);
        sort(nodes, first.x, first.y, x.length);
        System.out.print(2);
    }
    public static void findstate(node first, int[]x
            , int[]y, List<node> nodes){
        for(int i=0;i<x.length;i++){
            //find less 2 node
            int dis[]= new int [x.length];
            for(int j= 0;j < x.length;j++){
                dis[j]= Math.abs(x[j] - x[i])
                        + Math.abs(y[j] - y[i]);
            }
            int small= 9999999;
            int f= 0;
            int s= 0;
            int t= 0;
            for(int j= 0; j< dis.length; j++){
                if(dis[j]< small){
                    small= dis[j];
                    f= j;
                }
            }
            small= 9999999;
            for(int j= 0; j< dis.length; j++){
                if(dis[j]< small&& j!= f){
                    small= dis[j];
                    s= j;
                    if(nodes.size()==0){
                        small= dis[j];
                        s= j;
                    }
                }
            }
            small=9999999;
            for(int j= 0;j<dis.length;j++){
                if(dis[j]<small&& j!= f&& j!= s){
                    small= dis[j];
                    t= j;
                    if(nodes.size()== 0){
                        small= dis[j];
                        t= j;
                    }
                }
            }
```

```
            node n=new node();
            n.x=x[i];
            n.y=y[i];
            n.index=i;
            n.next=new node();
            n.next.x=x[s];
            n.next.y=y[s];
            n.next.index=s;
            n.next.prev=n;
            n.prev=new node();
            n.prev.x=x[t];
            n.prev.y=y[t];
            n.prev.index=t;
            n.prev.next=n;
            nodes.add(n);
            //s value=
        }
    }
    public static void findrandom(node first, int[]x, int[]y
            , List<node> nodes){
        int min= 999999999;
        int big= 0;
        for(int i= 0; i<x.length; i++){
            node first_copy= new node();
            first_copy.x= first.x;
            first_copy.y= first.y;
            first_copy.used= first.used;
            first_copy.total_dis= first.total_dis;
            first_copy.next= first.next;
            first_copy.prev= first.prev;
            int find= find(first_copy,i,x,y);
            if(find== 0){
                te++;
                node next= new node();
                next.x= x[i];
                next.y= y[i];
                int dis= Math.abs(x[i] - first.x)
                        + Math.abs(y[i] - first.y);
                next.total_dis= first_copy.total_dis + dis;
                next.used= first_copy.used+1;
                first_copy.next= next;
                next.prev= first_copy;
                first_copy= first_copy.next;
                int findSmall= findSmall(first_copy,x,y);
                if(findSmall== 1){
                    nodes.add(first_copy);
                }
```

```
                int cc= 0;
                if(first_copy.total_dis>big){
                    cc+= 1;
                    big= first_copy.total_dis;
                }else if(first_copy.total_dis< min){
                    cc+= 1;
                    min= first_copy.total_dis;
                }
                if(cc>0)
                    findrandom(first_copy, x, y,nodes);
            }
        }
    }
    public static void sort( List<node> nodes,int x, int y,int N){
        if(N> 41){
            for(int i= 0; i< nodes.size(); i++){
                System.out.println(nodes.get(i).prev.x+ ":"+nodes.get(i).prev.y+
                        "<-"+nodes.get(i).x+ ":"+ nodes.get(i).y+ "->"
                        + nodes.get(i).next.x+ ":"+nodes.get(i).next.y);
            }
            return;
        }
        node temp= nodes.get(nodes.size()-1);
        node temp1= nodes.get(nodes.size()-1);
        System.out.println("the short one");
        node []out= new node[N];
        while(temp!= null){
            out[temp.used]= temp;
            temp= temp.prev;
        }
        while(temp1!= null){
            out[temp1.used]= temp1;
            temp1= temp1.next;
        }
        for(int i1= 0; i1< out.length; i1++){
            if(out[i1]!= null){
                System.out.println("x:"+ out[i1].x
                        + "y:"+out[i1].y);
            }
        }
        System.out.println("the shortest distance:"
                + out[out.length-1].total_dis);
    }
    public static int find(node first, int i, int[] x, int[] y){
        if(first.x== x[i]&& first.y== y[i]){
            return 1;
        }
```

```
        node temp= first;
        node temp1= first;
        while(temp.prev!= null){
            temp= temp.prev;
            if(temp.x== x[i]&& temp.y== y[i]){
                return 1;
            }
            if(temp.x==x[x.length-1]&& temp.y==y[y.length-1]){
                if(temp.used!=x.length){
                    return 1;
                }
            }
            if(temp.x==x[0]&& temp.y==y[0]){
                if(temp.used != 0){
                    return 1;
                }
            }
        }
        while(temp1.next!= null){
            temp1= temp1.next;
            if (temp1.x== x[i]&& temp1.y== y[i]) {
                return 1;
            }
            if(temp1.x== x[x.length- 1]&& temp1.y== y[y.length- 1]){
                if(temp1.used!= x.length){
                    return 1;
                }
            }
            if(temp1.x== x[0]&& temp1.y== y[0]){
                if(temp1.used!= 0){
                    return 1;
                }
            }
        }
        if(temp.used!= 0){
            return 1;
        }
        if(temp.x!=x[0]|| temp.y!=y[0]){
            return 1;
        }
        return 0;
    }
    public static int findSmall(node first, int []x, int []y){
        while(first.next!=null) {
            first= first.next;
        }
        if(first.x== x[x.length-1]&& first.y== y[y.length-1]){
```

```
            if(first.used== x.length- 1){
                if(first.total_dis<small){
                    small= first.total_dis;
                    return 1;
                }
            }
        }
        return 0;
    }
    public static void findpath(node first, int[]x
            , int[]y, List<node> nodes){
        int min= 999999999;
        int big= 0;
        for(int i= 0; i< x.length; i++){
            int find= find(first, i, x, y);
            if(find== 0) {
                int dis= Math.abs(x[i]- first.x)
                        + Math.abs(y[i]- first.y);
                int total_dis= first.total_dis+ dis;
                if (total_dis>= big) {
                    big= total_dis;
                }
                if (total_dis<= min) {
                    min= total_dis;
                }
            }
        }
        for(int i= 0; i< x.length; i++){
            node first_copy= new node();
            first_copy.x= first.x;
            first_copy.y= first.y;
            first_copy.used= first.used;
            first_copy.total_dis= first.total_dis;
            first_copy.next= first.next;
            first_copy.prev= first.prev;
            int find= find(first_copy, i, x, y);
            if(find== 0){
                te++;
                node next= new node();
                next.x= x[i];
                next.y= y[i];
                int dis= Math.abs(x[i]- first.x)
                        + Math.abs(y[i]- first.y);
                next.total_dis= first_copy.total_dis+ dis;
                next.used= first_copy.used+ 1;
                first_copy.next= next;
                next.prev= first_copy;
```

```
                first_copy= first_copy.next;
                int findSmall= findSmall(first_copy, x, y);
                if(findSmall== 1){
                    nodes.add(first_copy);
                }
                int cc= 0;
                if(first_copy.total_dis>= big*1){
                    cc+= 1;
                    //big=first_copy.total_dis;
                }else if(first_copy.total_dis<= min*1){
                    cc+= 1;
                    //min=first_copy.total_dis;
                }
                if(cc> 0) {
                    findpath(first_copy, x, y,nodes);
                }
            }//
        }
    }
}
```

```
package org.deta.tinos.txt;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
public class TXTSwap{
    //注意字节编码的读写正确
    public static Object[][] txtFileToObjectMatrix(String filePath
            , String stopBy) throws IOException {
        Object[][] object= new Object[65535][];
        InputStream inputStream= new FileInputStream(new File(filePath));
        BufferedReader cReader= new BufferedReader(
                new InputStreamReader(inputStream));
        String ctempString= null;
        int i= 0;
        while (null!= (ctempString= cReader.readLine())) {
            object[i++]=ctempString.split(stopBy);
        }
        cReader.close();
        return object;
    }
    public static Object[][] txtFileToObjectMatrixWithRange(String filePath
```

```
            , String stopBy, int rangeBegin, int rangeEnd) throws IOException {
        Object[][] object= new Object[rangeEnd- rangeBegin][];
        InputStream inputStream= new FileInputStream(new File(filePath));
        BufferedReader cReader= new BufferedReader(
                new InputStreamReader(inputStream));
        String ctempString= null;
        int i= 0;
        while (null!= (ctempString= cReader.readLine())) {
            i++;
            if(i<=rangeEnd&& i>= rangeBegin) {
                object[i++]=ctempString.split(stopBy);
            }
        }
        cReader.close();
        return object;
    }
    public static List<String> txtFileToListStringWithRange(String filePath
            , String stopBy, int rangeBegin, int rangeEnd) throws IOException {
        List<String> list= new ArrayList<>();
        InputStream inputStream= new FileInputStream(new File(filePath));
        BufferedReader cReader= new BufferedReader(
                new InputStreamReader(inputStream));
        String ctempString= null;
        int i= 0;
        while (null!= (ctempString= cReader.readLine())) {
            i++;
            if(i<=rangeEnd&& i>=rangeBegin) {
                list.add(ctempString);
            }
        }
        cReader.close();
        return list;
    }
    public static List<String> txtFileToListString(String filePath)
            throws IOException {
        List<String> list= new ArrayList<>();
        InputStream inputStream= new FileInputStream(new File(filePath));
        BufferedReader cReader= new BufferedReader(
                new InputStreamReader(inputStream));
        String ctempString= null;
        while (null!= (ctempString= cReader.readLine())) {
            list.add(ctempString);
        }
        cReader.close();
        return list;
    }
    public static List<String[]> txtFileToListStringArray(String filePath
```

```
            , String stopBy) throws IOException {
        List<String[]> list= new ArrayList<>();
        InputStream inputStream= new FileInputStream(new File(filePath));
        BufferedReader cReader= new BufferedReader(
                new InputStreamReader(inputStream));
        String ctempString= null;
        while (null!= (ctempString= cReader.readLine())) {
            list.add(ctempString.split(stopBy));
        }
        cReader.close();
        return list;
    }
}
```

```
package org.deta.tinos.vector;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import org.deta.tinos.stable.Stable;
import com.google.gson.Gson;
public class VectorSwap{
    public static Map<String, Object> vectorToMap(
            Vector<Object> vector){
        Map<String, Object> map= new LinkedHashMap<>();
        Iterator<Object> iterator= vector.iterator();
        int i= 0;
        while(iterator.hasNext()) {
            map.put(Stable.STRING_EMPTY+ i++, iterator.next());
        }
        return map;
    }
    public static List<Object> vectorToList(Vector<Object> vector){
        List<Object> list= new ArrayList<>();
        Iterator<Object> iterator= vector.iterator();
        while(iterator.hasNext()) {
            list.add(iterator.next());
        }
        return list;
    }
    public static Object[] vectorToArray(Vector<Object> vector){
        return vector.toArray();
    }
```

```
    public static Iterator<Object> vectorToIterator(
            Vector<Object> vector){
        return vector.iterator();
    }
    public static String vectorToJsonString(Gson gson
            ,Vector<Object> vector){
        return gson.toJson(vector);
    }
    public static Set<Object> vectorToSet(Vector<Object> vector){
        Set<Object> sets= new TreeSet<>();
        Iterator<Object> iterator= vector.iterator();
        while(iterator.hasNext()) {
            sets.add(iterator.next());
        }
        return sets;
    }
}
```

```
package org.deta.tinos.xml;
import org.json.XML;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import java.util.List;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.Hashtable;
import org.json.JSONArray;
import org.json.JSONObject;
import com.google.gson.Gson;
import org.deta.tinos.json.JsonSwap;
import com.google.gson.reflect.TypeToken;
public class XMLSwap{
    public static List<Object> xmlToList(String string, String key) {
        JSONArray jSONArray= XML.toJSONObject(string).getJSONArray(key);
        List<Object> list= new ArrayList<>();
        for(int i= 0; i< jSONArray.length(); i++){
            Object object= jSONArray.get(i);
            if(object instanceof JSONObject){
                list.add(new Gson().fromJson(jSONArray.getJSONObject(i).toString()
                        , new TypeToken<Map<String, Object>>(){}.getType()));
            }else if(object instanceof String){
                list.add(String.valueOf(object));
            }else if(object instanceof JSONArray){
                list.add(JsonSwap.jsonArrayToList(jSONArray.getJSONArray(i)));
            }
```

```
        }
        return list;
    }
    public static Vector<Object> xmlToVector(String string, String key) {
        JSONArray jSONArray= XML.toJSONObject(string).getJSONArray(key);
        Vector<Object> vector= new Vector<>();
        for(int i= 0; i< jSONArray.length(); i++){
            Object object= jSONArray.get(i);
            if(object instanceof JSONObject){
                vector.add(new Gson().fromJson(jSONArray.getJSONObject(i).toString()
                        , new TypeToken<Map<String, Object>>(){}.getType()));
            }else if(object instanceof String){
                vector.add(String.valueOf(object));
            }else if(object instanceof JSONArray){
                vector.add(JsonSwap.jsonArrayToList(jSONArray.getJSONArray(i)));
            }
        }
        return vector;
    }
    public static Set<Object> xmlToSets(String string, String key) {
        JSONArray jSONArray= XML.toJSONObject(string).getJSONArray(key);
        Set<Object> sets= new TreeSet<>();
        for(int i= 0; i< jSONArray.length(); i++){
            Object object= jSONArray.get(i);
            if(object instanceof JSONObject){
                sets.add(new Gson().fromJson(jSONArray.getJSONObject(i).toString()
                        , new TypeToken<Map<String, Object>>(){}.getType()));
            }else if(object instanceof String){
                sets.add(String.valueOf(object));
            }else if(object instanceof JSONArray){
                sets.add(JsonSwap.jsonArrayToList(jSONArray.getJSONArray(i)));
            }
        }
        return sets;
    }
    public Hashtable<String, Object> xmlToHashtable(String xmlString, Gson gson){
        JSONObject jSONObject= XML.toJSONObject(xmlString);
        Hashtable<String, Object> hashTable= gson.fromJson(jSONObject.toString()
                , (Type) new Hashtable<String, Object>());
        return hashTable;
    }
}
```