Introduction to Computer Programming (Java A)
Lab Exercise 9

[Objective]

- Learn how to read / write text files in JAVA
- Learn to create packages to organize classes

[Exercise]

Part 1: Read and write text file

1. Write a Java program to get a list of all file/directory names in a directory.

```
import java.io.File;


public class ListFolder {
    public static void main(String a[]) {
        File file = new File("C:\\Users\\");
        String[] fileList = file.list();
        for (String name : fileList) {
            System.out.println(name);
        }

    }
}
```

2. Write a Java program to read all the content from an input text file, convert all of them into uppercase letters, and write the result into an output file. The output file should be placed at the same folder as the input file. The input file is assumed to contain only English letters, numbers and punctuation marks.

```
import java.io.*;
public class ReadWriteTextFile
{
    public static void main(String[] args)
    {
        try
        {
            BufferedReader in=new BufferedReader(new
FileReader("TestIn.txt"));
            BufferedWriter writer=new BufferedWriter(new
FileWriter("TestOut.txt"));
            int temp;
            while ((temp=in.read())!=-1)
            {
```

```
                if (temp>='a' &&temp<='z')
                {
                    writer.write(temp-'a'+'A');
                }else
                {
                    writer.write(temp);
                }
            }
            in.close();
            writer.close();
        } catch (IOException e)
        {
            System.out.println("There is no this file!");
        }
    }
}
```

Tips: There are many ways to read and write text files. It is only an example here. And you may get more details from chapter 17 in the text book ***Java How to Program***.

3. Grammar correction tool

Kids always make mistake in English writing. A common mistake is that they always forget to use capital letter at the beginning of a sentence. Write a Java program to help them correct the mistakes.

Usually, the word after a full stop (.) is regarded as the beginning of a new sentence. However, a full stop (.) also means abbreviation. When a word is abbreviated after the first few letters, the traditional rule is to put a full stop after the abbreviation, for example, Dr., Mr. and FYI.. We can assume every abbreviation begins with a capital letter. This indicates whether the current stop means the end of sentence or an abbreviation.

The Java program should first read the text scanner, and then write the result to an text file.

Sample Input 1

today I borrow a book from my neighbor. he come and get it back tomorrow.

Sample Output 1

Today I borrow a book from my neighbor. He comes and gets it back tomorrow.

Sample Input 2

please get the report from BBC. news ASAP. as the boss want to read them now. he will not stay here until the evening.

Sample Output 2

Please get the report from BBC. news ASAP. as the boss want to read them now. He will not stay here until the evening.

Part 2: Package and classpath

Given the Circle.java file you wrote previously, add the following package declaration statement at the beginning of the .java file.

```
package sustech.cs102a.lab8;
```

Now go to the directory where the Circle.java file resides and run the following command to compile the Circle.java file. Observe what would happen.

```
javac Circle.java
```

You will find a Circle.class file appearing in your working directory. If you run the directory listing command or check in the directory window, you can see both the .java and .class files.

Now suppose you want to run the Circle class. You might wish to run this command:

```
java Circle
```

Unfortunately, you will get the following error message:

```
Exception in thread "main" java.lang.NoClassDefFoundError: Circle (wrong name: sustech/cs102a/lab9/Circle)
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClassCond(ClassLoader.java:631)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:615)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:141)
        at java.net.URLClassLoader.defineClass(URLClassLoader.java:283)
        at java.net.URLClassLoader.access$000(URLClassLoader.java:58)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:197)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
Could not find the main class: Circle.  Program will exit.
```

Or

```
错误: 找不到或无法加载主类 Cirle
```

This is because by adding a package declaration, the fully qualified name of the Circle class becomes “sustech.cs102a.lab8.Circle”. We cannot simply use the simple name to run the class.

It should be run the following command in the directory “sustech/cs102a/lab8” included.

```
java sustech.cs102a.lab8.Circle
```

Tips: the “javac” command may run in 2 ways:

(1) In directory “lab8”: `javac Cirle`

(2) In upper directory of "sustech": `javac sustech\cs102a\lab8`

In both ways, the "Circle.class" will be in the directory "sustech\cs102s\lab8".

Now, continue to create a CircleTest.java file with the following code in lab9:

```
package sustech.cs102a.lab9;
import sustech.cs102a.lab8.Circle;

public class CircleTest {
    public static void main(String[] args) {
        Circle c = new Circle(1.0, 0.0, 0.0);
        c.position();
    }
}
```

In the code, we need to import the sustech.cs102a.lab8.Circle class because it is declared in another package.

Note that for all the above steps, we assume that we don't change our working directory during the whole process. If we change to another directory and wish to run the CircleTest class from there, we need to specify the classpath as follows:

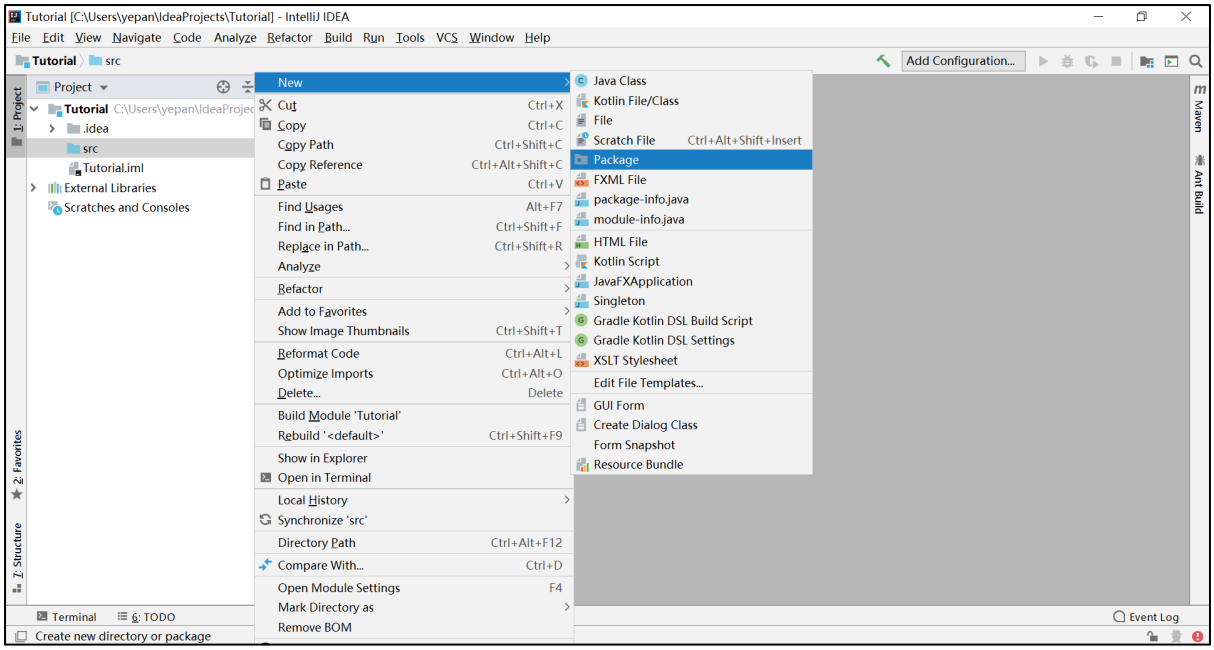
```
java -cp parent-dir-of-sustech sustech.cs102a.lab10.CircleTest
```

If the classpath contains several directories, we should use directory separators to separate them. Semicolon(;) is the directory separator on Windows. On Unix/Linux/Mac, you should use colon (:). Below is an example:

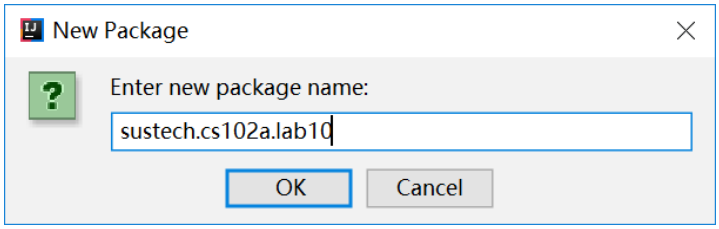
```
java -cp dir1;dir2;dir3 ClassToRun
```

The above tutorial explains package and classpath at a low level. In an IDE, creating packages and declaring classes in them is as easy as pie. We provide the steps below for IntelliJ IDEA.

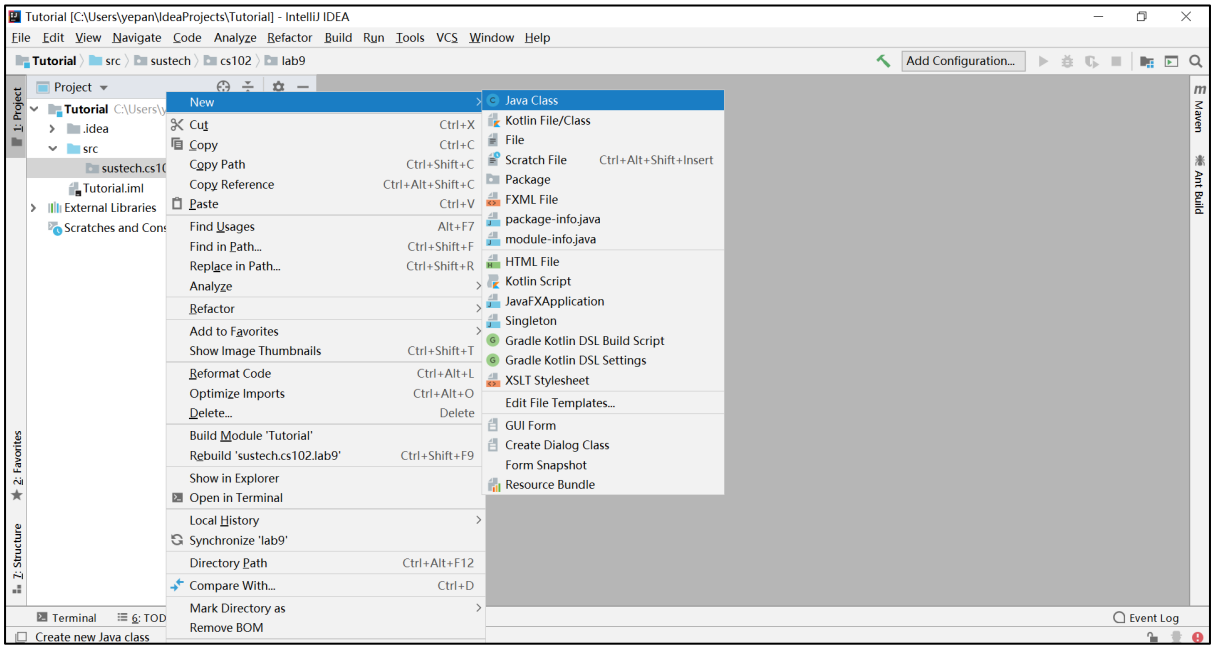
Step 1: In a project, right click on the src, then click New->Package



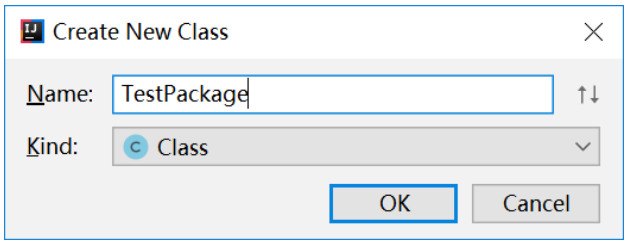
Step 2: Enter the package nam in the dialog box, click ok and you will see the package created.



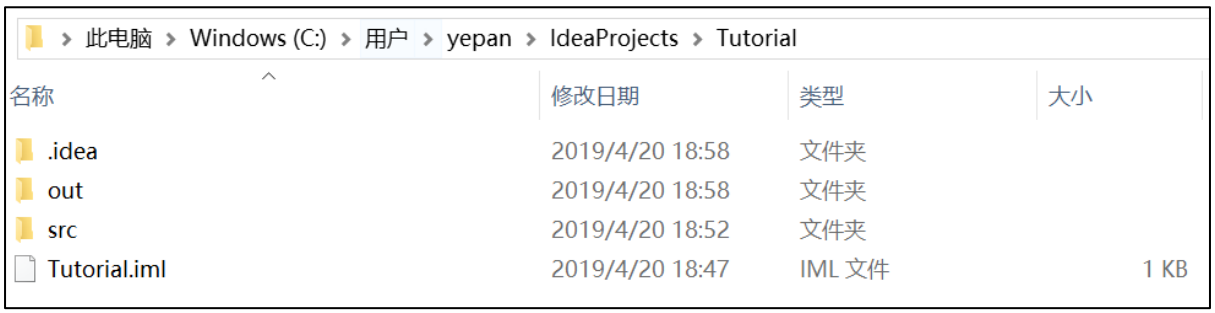
Step 3: Right click on the package, then click New->Java Class



Step 4: Enter the class name in the dialog box, click ok and you will see the skeleton code of the newly created class. You will find that the package declaration statement is added automatically.



Now compile the class and go to the directory where the project is stored. You will see that project directory contains two sub-directories: “src” stores the .java source files and “out” stores the .class files after compilation.



If you check the src and out directories, you will find that the TestPackage.java and TestPackage.class files reside in the following directories, respectively:

```
src/sustech/cs102a/lab9
out/production/[Project Name]/sustech/cs102a/lab9
```

IDEA helps manage all the stuff automatically. The way how source files and class files are organized may be different for other IDEs (for example, Eclipse), but the TestPackage.class file is always put under sustech/cs102a/lab9 after compilation.

4. To put the java file of “exercise 2” into package lab9, which folder should we put the file “Testin.txt” in?
(1) To run by the command “javac” and “java” in command lines.
(2) To run by IDEA.

Tips:
It is recommended to put java files of each lab class into different packages in the same project, but not to put all files into a directory in a project, or to create a project in each class.