面向对象程序设计
作业报告

第 二 次

姓名	曹家豪
班级	软件 2204 班
学号	2226114017
电话	13572763245
Email	caojiahao@stu.xjtu.edu.cn
日期	2023-11-6

目录

实验一

1、题目：创建一个表示复数的类

创建一个用表示复数的 Complex 类类型，当创建完这个类以后，就可以直接使用这个数据类型完成需要复数参与的各种算法中，图 1 是对创建的 Complex 类型的具体说明。

该题需要完成的任务如下：

任务 1：编写 Complex 类类型。

任务 2：编写一个测试程序（TestComplex），要求对 Complex 类型中的每一个公共实例方法进行测试。

任务 3：编写一个使用 Complex 的简易客户端程序（ComplexApp），该程序完成提示用户输入两个复数，之后按照如下输出样式完成对这两个复数完成操作的相关调用。

Enter complex number 1 (real and imaginary part): 1.1 2.2

Enter complex number 2 (real and imaginary part): 3.3 4.4

Number 1 is: (1.1 + 2.2i)

(1.1 + 2.2i) is NOT a pure real number

(1.1 + 2.2i) is NOT a pure imaginary number

Number 2 is: (3.3 + 4.4i)

(3.3 + 4.4i) is NOT a pure real number

(3.3 + 4.4i) is NOT a pure imaginary number

(1.1 + 2.2i) is NOT equal to (3.3 + 4.4i)

(1.1 + 2.2i) + (3.3 + 4.4i) = (4.4 + 6.6000000000000005i)

2、数据设计

该实验共设计 Complex、ComplexTest 与 ComplexApp 三个类

Complex 类根据题干中 UML 图定义了 real、imag 数据。

方法包含两种构造方法，还有 getReal()、getImag()、setReal()、setImag()、setValue()等方法以获取/设定复数，equal（）、isReal（）等函数进行复数的判断，add（）、subtract（）等函数进行复数的加减乘除等操作。

此外为实现 double 类型数据的相等操作，设定了数据变量 EPSION。

3、个别算法设计

①定义|a-b|<=EPSION　即认定 a=b;

②在已有安排判断 double 类型数据相等的基础上，判断复数类型 equal 的证据则为对应实、虚部相等;

③此外，判断实/虚部是否等于 0.0，即可说明该复数是否为纯虚/实数。

④针对 double 中的 double.NaN 型舒数据，使用 if 语句以及 Double.isNaN 方法来判断并确保返回值合理

4、主干代码说明

（1）复数乘法的实现:

通过自行运算可得 $(a+bJ) * (c+dJ) = (ac-bd) + (ad+bc)J$

因此直接写出函数 myltiply（）；

```
1. public Complex multiply(Complex right){
2.         Complex rescomplex=new Complex(this.real*right.getR
   eal()-
   this.imag*right.getImag(),this.imag*right.getReal()+this.re
   al*right.getImag());//实现公式
3.         return rescomplex;
4.     }
```

（2）复数除法的实现

自行运算可得$\frac{a+bj}{c+dj} = \frac{(ac+bd)+(bc-ad)j}{c^2+d^2}$，定义分母为 dem=$c^2+d^2$ 分别计算实部 factor1=$\frac{(ac+bd)}{c^2+d^2}$，虚部 factor2=$\frac{(bc-ad)}{c^2+d^2}$，利用 setValue 创建新复数并返回。据此写出 divide 函数

此外，利用 Double.isNaN 方法判断得到 double 型数据是否合理

```
1. public Complex divide(Complex right){
2.         double dem=right.getReal()*right.getReal()+right.ge
   tImag()*right.getImag();//计算分母
3.         double factor1=(this.real*right.getReal()+this.imag
   *right.getImag())/dem;//计算实部
4.         if (Double.isNaN(factor1)) {
5.             System.out.println("The value is NaN");
6.         }//判断实部是否合理
7.         double factor2=(this.imag*right.getReal()-
   this.real*right.getImag()/dem);//计算虚部
8.         if (Double.isNaN(factor2)) {
9.             System.out.println("The value is NaN");
10.        }//判断虚部是否合理
11.        Complex rescomplex=new Complex(factor1,factor2);
12.        return rescomplex;//返回新创建复数
13.    }
14.}
15.
```

(3) 复数距离原点距离的实现

对应坐标变换可知，distance $(a+bj)=\sqrt{a^2+b^2}$,据此实现 getDistance 函数：

```
1. public double getDistance(){
2.         return Math.sqrt(this.real*this.real+this.imag*this
   .imag);
3.     }//对应坐标变换可知，distance=√(a²+b²)
```

5、运行结果展示

(1)ComplexTest

测试程序在附录中，结果对应如下：

```
"E:\New Folder\bin\java.exe" "-javaagent:E:\java环境\IntelliJ I
true
false
2.0
4.0
3.0+4.0j
false
true
5.0
6.0+8.0j
0.0+0.0j
-7.0+24.0j
1.0+11.52j

Process finished with exit code 0
```

(2)ComplexApp

```
"E:\New Folder\bin\java.exe" "-javaagent:E:\java环境\IntelliJ IDEA Community Edition 2023.2.1\lib
Enter complex number 1 (real and imaginary part): 1.1 2.2
Enter complex number 2 (real and imaginary part): 3.3 4.4
Number 1 is: 1.1+2.2j
1.1+2.2j is NOT a pure real number
1.1+2.2j is NOT a pure imaginary number
Number 2 is: 3.3+4.4j
3.3+4.4j is NOT a pure real number
3.3+4.4j is NOT a pure imaginary number
1.1+2.2j is NOT equal to (3.3+4.4j)
(1.1+2.2j)+(3.3+4.4j)=4.4+6.6000000000000005j

Process finished with exit code 0
```

实验 2

1、题目：使用 Complex 类型绘制分形图

1973 年，曼德布罗特（B.B.Mandelbrot）在法兰西学院讲课时，首次提出了分维和分形几何的设想。分形（Fractal 一词，是曼德布罗特创造出来的，其原意具有不规则、支离破碎等意义， 分形几何学是一门以非规则几何形态为研究对象的几何学。由于不规则现象在自然界是普遍存在 的，因此分形几何又称为描述大自然的几何学。分形几何建立以后，很快就引起了许多学科的关 注，这是由于它不仅在理论上，而且在实用上都具有重要价值。

现在，使用刚刚创建的 Complex 类型和课堂上的 Picture 类型，完成一个分形图的制作。 此次制作分形图片的主角是 JuliaSet（朱利亚集合），该集合是由复平面上的若干个点构成， 以法国数学家加斯顿·朱利亚（Gaston Julia）的名字命名。 设定一个常复数 c，对于复平面上的某个点 z0 完成如下公式计算：zn+1 = (zn) 2+

c，反复迭代 之后，得到一个序列：z1，z2，…，如果这个序列发散于无穷大，那么 z0 这个数就不在朱利亚集合

中，如果这个序列最终在某一范围内收敛于某一值，那么 z0 这个数就在朱利亚集合中。将属于朱利 亚集合中的点着黑色，不在朱利亚集合中的点着为不同等级的灰色（最终是白色），那么一副分形图就制作出来了。

2、数据设计

实验使用 MakeJuliaSet 与 Picture 两个类，在 Picture 类中定义了图像的创建、颜色设定、保存等操作，MakeJuliaSet 则借用这些操作绘出图形。

在 MakeJuliaSet 类中，定义方法 toSwitch（）和 extentOfDivergency（），其中 toSwitch（）用于将图像 1024× 1024 的坐标转换为复数域（-2，2）的坐标；extentOfDivergency（）用于返回对应坐标值按茱莉亚集合处理方法后的发散程度。

3、算法设计

①创建空白图片，用双层循环遍历图片的每个像素点

②完成像素点的坐标转换

③判断该坐标点的发散程度

④根据发散程度返回值设定像素点颜色

⑤遍历结束后，存储图像

4、主干代码说明

(1)坐标转换

```
1. public Complex toSwitch(int x,int y){
2.     double a=(x-511.5)/255.75;
3.     double b=(511.5-y)/255.75;
4.     Complex c=new Complex(a,b);
5.     return c;
6.     }//自行绘图完成1024×1024到4×4的转换
```

(2)发散程度判断

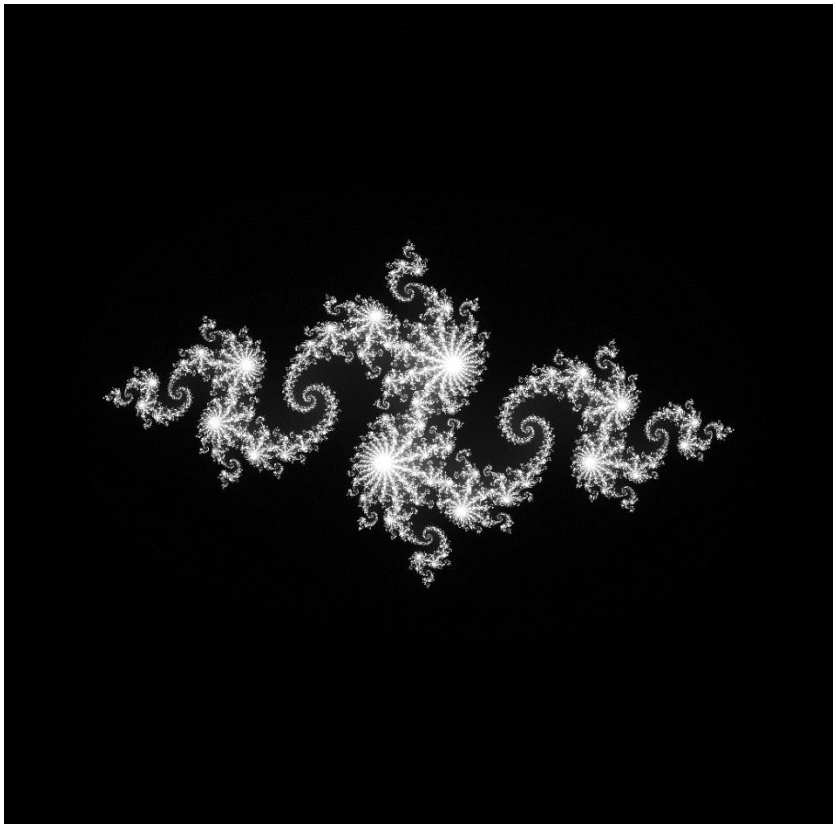
//根据题目所给内容可直接写出

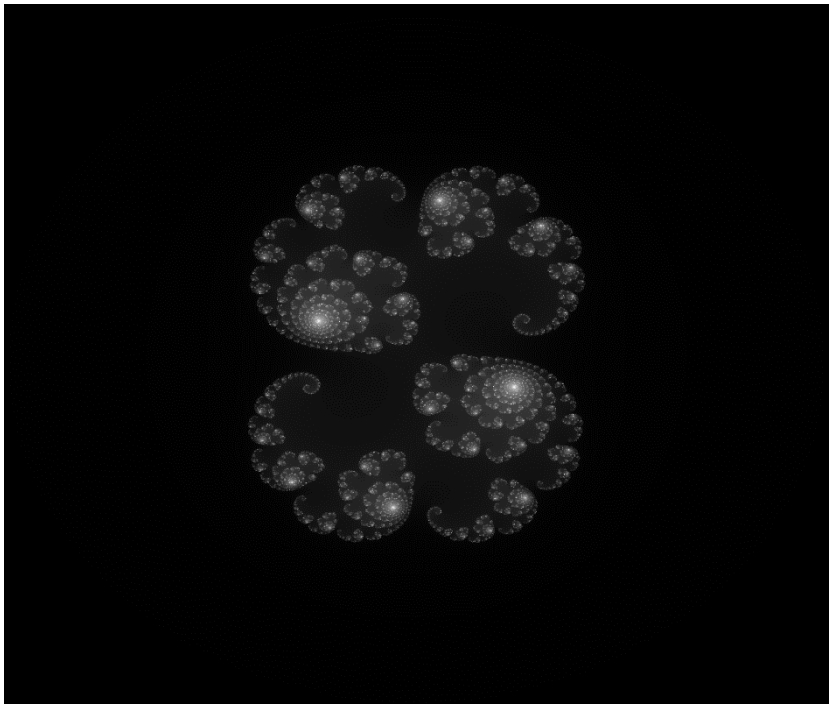
```
1. public int extentOfDivergency(Complex c,Complex z0){
2.         Complex z=z0;
3.         int max=255;
4.         for(int t=0;t<max;t++){
5.             if(z.getDistance()>2) return t;
6.             z=z.multiply(z).add(c);
7.         }
8.         return max;
9.     }
```

(3)设定像素点颜色

```
1. for(int i=0;i<1024;i++){
2.             for(int j=0;j<1024;j++){//逐个遍历
3.                 int degree=key.extentOfDivergency(c,key.toS
   witch(i,j));//获取发散程度
4.                 Color color=new Color(degree,degree,degree)
   ;//利用发散程度设定对应颜色
5.                 JuliaSet.setColor(i,j,color);
6.             }
7.         }
```

5、运行结果展示





6、总结和收获

(1)掌握 Picture 类的使用，学习了图像的创建、颜色设置、保存方法

⑵被 static 的方法内部无法直接调用同一个类内的函数，需要新创建类对象

⑶学习了在函数中利用迭代的方法

实验 3

1、题目：模拟一个购物车

双 11 即将到来，购物车满起来!!! 创建一个购物车类类型（ShoppingCart），每一个 ShoppingCart 对象应该包含若干个商品（Item 类型）对象，购物车类型应该至少提供添加商品、删除商品以及汇总购物车中商品价格的能 力。这是一个开放题目，发挥的空间很大。下面是对该题完成所需满足的基本要求:

要求 1：商品类型至少要有三种以上（不同类型的商品有不同的属性），为了复用和多态， 请使用继承完成商品这一组类型的构建;

要求 2：不论是商品类型，还是购物车类型都需要实现 toString 方法，针对不同类型中包含 的不同数据成员订制输出内容;

要求 3：购物车对象应该不限制包含商品的数量，这时需要使用具有可动态扩容能力的数 组数据类型（ArrayList），这个类型是 Java API 中已经定义好的数据类型，具体使用方法参见本文的 附录。

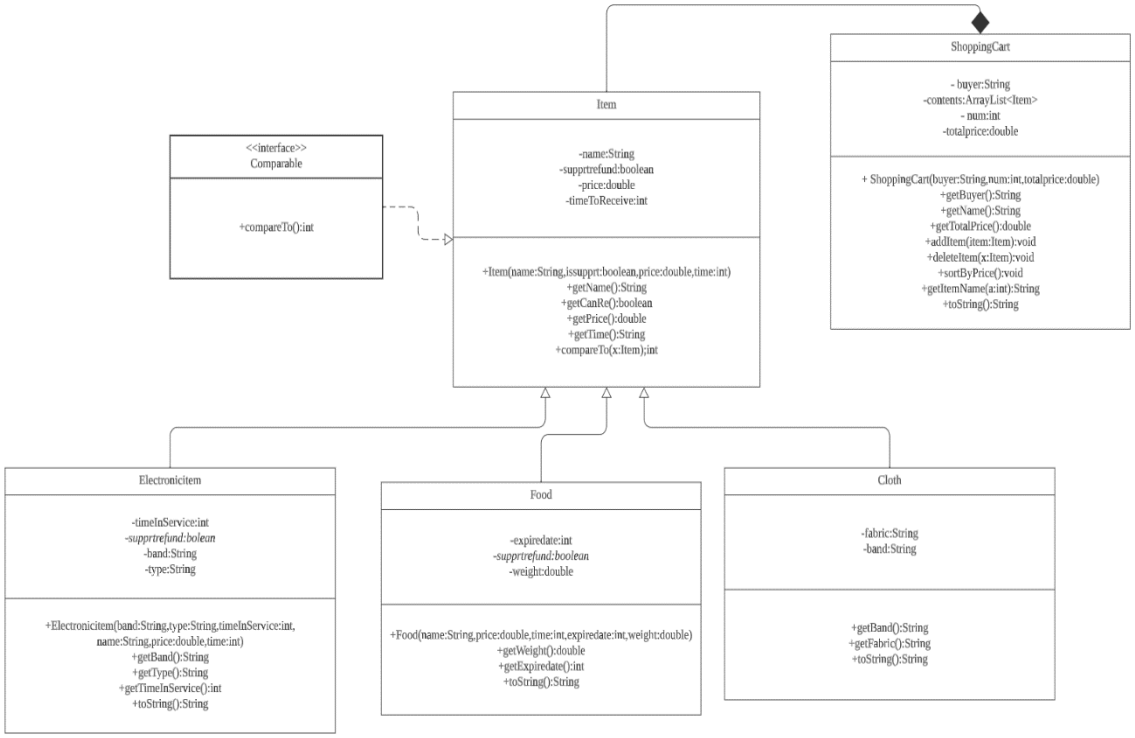
要求 4：购物车的商品要能排序，因此要求商品类必须实现 Comparable 接口，排序算法可 以使用冒泡排序算法。商品排序的规则自行规定，需要在实验报告中说明。

要求 5：该题在实验报告中的设计环节描述请使用 UML 类图完成。

要求 6：编写一个客户端程序，要有商品添加、商品删除、商品价格汇总以及商品排序等 环节的调用和结果展示。

2、数据与算法设计

UML 图如下：



(1)实验中完成了对 ShoppingCart、Item、Electronicitem、Cloth、Food 等类，其中 ShoppingCart 与 Item 为一对多的组合关系，Item 类实现了 Comparable 接口中 compareTo 方法的实现，而 Cloth、Food、Electronicitem 实现了对 Item 类的继承

(2)通过 ShoppingCart 类中 Array List<Item>类型的定义实现组合关系

⑶针对不同类型设定了对应的 toString（）方法。

⑷此外，设立了不同类型对应的价格、保质期、保修期等属性，以及对应获取方法

3、主干代码说明

⑴用 ArrayList<Item>存储 Item;

```
1. import java.util.ArrayList;
2. import java.util.Iterator;
3.     private ArrayList<Item> contents;//用ArrayList 存储物品
```

⑵添加商品

```
1. public void addItem(Item item) {
2.         contents.add(item);
3.         totalprice += item.getPrice();
4.         num++;
5.     }//添加商品
```

⑶删除商品

使用 Iterator 类遍历 comtens 数组，找到同名商品后后用 move（）方法删除，同时商品数以及总价格减去对应值

```
1. public void deleteItem(Item x) {
2.         Iterator<Item> iterator = contents.iterator();
3.         while (iterator.hasNext()) {
4.             Item item = iterator.next();
5.             if (item.getName().equals(x.getName())) //字符串
   比较{
6.                 iterator.remove();
7.                 num--;
8.                 totalprice -= item.getPrice();
9.                 break;
10.            }
11.        }
12.    }//删除商品
```

⑷按价格排序

使用冒泡排序法，将 contents 数组按照商品价格从小到大排序，利用 Item 类的 compareTo 方法实现相邻两个商品价格的比较

```
1. public void sortByPrice() {
2.         for (int i = 0; i < num - 1; i++) {
3.             for (int j = 0; j <num-1-i; j++) {
4.                 if (contents.get(j).compareTo(contents.get(
   j+1))==1) {
5.                     Item item = contents.get(j);
6.                     contents.set(j, contents.get(j + 1));
7.                     contents.set(j + 1, item);
8.                 }
9.             }
10.         }
11.     }//用冒泡排序法将商品按价格从小到大进行排序
```

(5)Item 中实现 Comparable 接口

实现 compareTo 函数，如果前者价格小于后者，返回 0；否则返回 1

```
1. package Practice3forood;
2. public abstract class Item implements Comparable<Item>/*实现
3.     public int compareTo(Item x){
4.         if(price>=x.getPrice())
5.             return 1;
6.         else return 0;
7.     }//实现 compareTo 函数，如果前者价格小于后者，返回 0；否则返
   回 1
```

4、运行结果展示

对应 ShoppingCartTest 的代码在附录中。

5、总结和收获

⑴更加熟悉掌握构造方法的定义和使用，以及在子类中如何显式地调用父类构造方法，如果父类中没有无参构造方法，则必须在子类中显式地调用。

⑵尝试了接口的使用，并完成了对实现接口函数的调用。

实验 4

题目：撰写继承、多态和接口方面的知识梳理

关于继承（inheritance）:

每个类（Object 类除外）都一定有且仅有一个直接继承的父类，继承表示了一种子类与父类之间“is a”的关系，即子类仍属父类的一种，但是是对父类的特殊化，而父类则是其所有子类的公共属性，基于这种关系，子类会继承父类中的所有属性与方法，为了实现子类的某些特殊性，子类又可以在其内部重写父类中的属性和方法，其中原来的属性会被隐藏，而原方法则会被改写。

一些知识点:

1、在继承构造方法时，如果没有显式定义构造方法，子类会直接继承父类的无参构造方法，而若父类没有无参构造方法，则需要在子类的构造方法定义中显示地调用;

2、重写父类方法时，须保持方法签名（包括方法名称，参数列表）的一致，且返回类型是父类返回类型的子类或相同；

3、由方法的继承引出四种修饰符，分别为 private（仅限本类）、default（限同 package）、protected（限同 package 或继承关系）、public（公共）；在方法的重写中，重写方法的限制性不能比父类被重写的方法更高；

4、this 与 super，可以通过这两个关键字分别调用自身和父类的方法乃至数据，它们可以作为子类构造方法的第一行使用，如果构造方法中第一行没有调用其他方法，那么 super（）将会被自动加入；

5、this 与 super 调用可以用于实例方法和构造方法，但无法在类方法（static 修饰方法）中使用；

6、abstract 关键字，由于多个子类有对同一方法的不同需求，但在父类中难以进行统一的定义，为了实现继承，提高代码的复用率，引入 abstract 关键字修饰符，意味着被修饰的方法可以只声明而不被具体定义；而含有 abstract 方法的类也成为 abstract 类；

关于接口（interface）

1、顺着 abstract 的概念，接口可以说是与类同级别的，且由一些抽象类方法组成的集合；

2、通过接口的使用，我们可以实现类的多重继承；

3、接口没有构造方法，不能被直接实例化，需要通过 implements 关键字被类实现，且实现一个接口必须实现该接口中所有方法；

4、实现接口中的方法需要显示地用 public 修饰；

关于多态（polymorphism）

经过对继承与实现接口的介绍，我们了解到这二者在面向对象程序设计中对于代码复用率的提高，而通过继承中对方法的重写，以及子类对接口的不同实现，我们由实现了在保证复用率的同时代码的变异，即完成不同需求，允许在继承的程序中出现同名方法，这就是多态，可以说继承是多态的基础，多态同样提高了对代码的复用。

附录：（每个题的源代码）

题目一：

Complex.java

```
4. package practice1forood;
5.
6. import static java.lang.Math.abs;
7. public class Complex {
8.     private double real=0.0;
9.     private double imag=0.0;
10.    final  double EPSILON=1e-8;
11.    public Complex(){
12.        this(0.0,0.0);
13.    };
14.    public Complex(double real,double imag){
15.        this.real=real;
16.        this.imag=imag;
17.    }
18.    public double getReal(){
19.        return this.real;
20.    }
21.    public void setReal(double newreal){
```

```
22.         this.real=newreal;
23.     }
24.     public double getImag(){
25.         return this.imag;
26.     }
27.     public void setImag(double newimag) {
28.         this.imag = newimag;
29.     }
30.     public void setValue(double newreal,double newimag){
31.         this.real=newreal;
32.         this.imag=newimag;
33.     }
34.     public String toString(){
35.         return this.real+"+"+this.imag+"j";
36.     }
37.     public boolean isReal(){
38.         if (abs(this.imag)<=EPSILON)
39.             return true;
40.         else
41.             return false;
42.     }
43.     public boolean isImaginary(){
44.         if (abs(this.real)<=EPSILON)
45.             return true;
46.         else
47.             return false;
48.     }
49.     public boolean equals(Complex another){
50.         boolean result=equals(another.getReal(),another.get
    Imag());
51.         if (result==true)
52.             return true;
53.         else return false;
54.     }
55.     public boolean equals(double real,double imag){
56.         if(abs(this.real-real)<=EPSILON&&abs(this.imag-
    imag)<=EPSILON)
57.             return true;
58.         else return false;
59.     }
60.     public double getDistance(){
61.         return Math.sqrt(this.real*this.real+this.imag*this
    .imag);
62.     }
63.     public Complex add(Complex right){
```

```
64.         Complex rescomplex=new Complex(this.real+right.getR
    eal(),this.imag+right.getImag());
65.         return rescomplex;
66.     }
67.     public Complex subtract(Complex right) {
68.         Complex rescomplex = new Complex(this.real - right.
    getReal(), this.imag - right.getImag());
69.         return rescomplex;
70.     }
71.     public Complex multiply(Complex right){
72.         Complex rescomplex=new Complex(this.real*right.getR
    eal()-
    this.imag*right.getImag(),this.imag*right.getReal()+this.re
    al*right.getImag());
73.         return rescomplex;
74.     }
75.     public Complex divide(Complex right){
76.         double dem=right.getReal()*right.getReal()+right.ge
    tImag()*right.getImag();
77.         double factor1=(this.real*right.getReal()+this.imag
    *right.getImag())/dem;
78.         if (Double.isNaN(factor1)) {
79.             System.out.println("The value is NaN");
80.         }
81.         double factor2=(this.imag*right.getReal()-
    this.real*right.getImag()/dem);
82.         if (Double.isNaN(factor2)) {
83.             System.out.println("The value is NaN");
84.         }
85.         Complex rescomplex=new Complex(factor1,factor2);
86.         return rescomplex;
87.     }
88. }
89.
```

ComplexTest.java:

```
1. package practice1forood;
2. public class TestComplex {
3.     public static void main(String[] args) {
4.         Complex a = new Complex();
5.         Complex b = new Complex(2.0, 3.0);
6.         System.out.println(a.isReal());
7.         System.out.println(b.isImaginary());
8.         System.out.println(b.getReal());
9.         b.setReal(3.0);
```

```
10.         b.setImag(4.0);
11.         System.out.println(b.getImag());
12.         System.out.println(b.toString());
13.         System.out.println(a.equals(3.0, 4.0));
14.         a.setValue(3.0, 4.0);
15.         System.out.println(a.equals(b));
16.         System.out.println(a.getDistance());
17.         System.out.println(a.add(b).toString());
18.         System.out.println(a.subtract(b).toString());
19.         System.out.println(a.multiply(b).toString());
20.         System.out.println(a.divide(b).toString());
21.     }
22. }
```

ComplexApp.java:

```
1. package practice1forood;
2. import java.util.Scanner;
3. public class ComplexApp {
4.         public static void main(String[] args) {
5.             Scanner scanner = new Scanner(System.in);
6.             System.out.print("Enter complex number 1 (real
   and imaginary part): ");
7.             double real1 = scanner.nextDouble();
8.             double imaginary1 = scanner.nextDouble();
9.             Complex c1 = new Complex(real1, imaginary1);
10.            System.out.print("Enter complex number 2 (real
   and imaginary part): ");
11.            double real2 = scanner.nextDouble();
12.            double imaginary2 = scanner.nextDouble();
13.            Complex c2 = new Complex(real2, imaginary2);
14.            System.out.println("Number 1 is: " + c1);
15.            System.out.println(c1 + " is" + (c1.isReal() ?
   "" : " NOT") + " a pure real number");
16.            System.out.println(c1 + " is" + (c1.isImaginary
   () ? "" : " NOT") + " a pure imaginary number");
17.            System.out.println("Number 2 is: " + c2);
18.            System.out.println(c2 + " is" + (c2.isReal() ?
   "" : " NOT") + " a pure real number");
19.            System.out.println(c2 + " is" + (c2.isImaginary
   () ? "" : " NOT") + " a pure imaginary number");
20.            System.out.println(c1 + " is" + (c1.equals(c2)
   ? "" : " NOT") + " equal to " + "("+c2+")");
21.            Complex sum = c1.add(c2);
22.            System.out.println("("+c1+")+("+ c2+")="+ sum);
23.
```

```
24.         }
25.}
```

题目二：

Picture.java:

```
1. package Practice2forood;
2.
3. import javax.imageio.ImageIO;
4. import java.awt.*;
5. import java.awt.image.BufferedImage;
6. import java.io.File;
7. import java.io.IOException;
8. public class Picture {
9.     private BufferedImage image;
10.    public int width;
11.    private int height;
12.    public Picture(String filename) throws IOException {
13.        File file = new File(filename);
14.        image = ImageIO.read(file);
15.        width = image.getWidth();
16.        height = image.getHeight();
17.    }
18.    public Picture(int width, int height){
19.        this.width = width;
20.        this.height = height;
21.        image = new BufferedImage(width, height, BufferedIm
   age.TYPE_INT_RGB);
22.    }
23.    public int getWidth(){
24.        return width;
25.    }
26.    public int getHeight(){
27.        return height;
28.    }
29.    public void setColor(int col, int row, Color c){
30.        image.setRGB(col, row, c.getRGB());
31.    }
32.    public Color getColor(int col, int row){
33.        int rgb = image.getRGB(col, row);
34.        return new Color(rgb);
35.    }
36.    public void save(String filename) throws IOException {
```

```
37.         String suffix = filename.substring(filename.lastInd
    exOf('.')+1);
38.         if (suffix.equalsIgnoreCase("png") || suffix.equals
    IgnoreCase("jpg"))
39.             ImageIO.write(image, suffix, new File(filename)
    );
40.     }
41.     public void darker(){
42.         for (int i = 0; i < width; i++)
43.             for (int j = 0; j < height; j++){
44.                 Color c = getColor(i, j);
45.                 setColor(i, j, c.darker());
46.             }
47.     }
48. }
```

MakeJuliaSet.java:

```
8.  package Practice2forood;
9.
10. import practice1forood.Complex;
11. import java.awt.*;
12. import java.io.IOException;
13. import java.util.Scanner;
14. public class MakeJuliaSet {
15.     public Complex toSwitch(int x,int y){
16.     double a=(x-511.5)/255.75;
17.     double b=(511.5-y)/255.75;
18.     Complex c=new Complex(a,b);
19.     return c;
20.     }
21.     public int extentOfDivergency(Complex c,Complex z0){
22.         Complex z=z0;
23.         int max=255;
24.         for(int t=0;t<max;t++){
25.             if(z.getDistance()>2) return t;
26.             z=z.multiply(z).add(c);
27.         }
28.         return max;
29.     }
30.     public static void main(String[] args) throws IOExcepti
    on {
31.         MakeJuliaSet key=new MakeJuliaSet();
32.         Picture JuliaSet=new Picture(1024,1024);
33.         System.out.printf("请输入 c 的实部与虚部: ");
```

```
        Scanner scanner=new Scanner(System.in);
        double real=scanner.nextDouble();
        double imag=scanner.nextDouble();
        Complex c=new Complex(real,imag);
        for(int i=0;i<1024;i++){
            for(int j=0;j<1024;j++){
                int degree=key.extentOfDivergency(c,key.toSwitch(i,j));
                Color color=new Color(degree,degree,degree);
                JuliaSet.setColor(i,j,color);
            }
        }
        JuliaSet.save("JuliaPic.png");
    }
}

```

题目三：

ShoppingCart.java:

```
package Practice3forood;
import java.util.ArrayList;
import java.util.Iterator;
public class ShoppingCart {
    private String buyer;//购买用户
    private ArrayList<Item> contents;//用ArrayList 存储物品
    private int num = 0;//商品数目
    private double totalprice = 0;//总价
    public ShoppingCart(String buyer, int num, double totalprice) {
        this.buyer = buyer;
        this.num = num;
        this.totalprice = totalprice;
        contents=new ArrayList<>();
    }
    public String getBuyer() {
        return buyer;
    }
    public int getNum() {
        return num;
    }
    public double getTotalPrice() {
        return totalprice;
```

```
28.     }
29.     public void addItem(Item item) {
30.         contents.add(item);
31.         totalprice += item.getPrice();
32.         num++;
33.     }//添加商品
34.     public void deleteItem(Item x) {
35.         Iterator<Item> iterator = contents.iterator();
36.         while (iterator.hasNext()) {
37.             Item item = iterator.next();
38.             if (item.getName().equals(x.getName())) {
39.                 iterator.remove();
40.                 num--;
41.                 totalprice -= item.getPrice();
42.                 break;
43.             }
44.         }
45.     }//删除商品
46.     public void sortByPrice() {
47.         for (int i = 0; i < num - 1; i++) {
48.             for (int j = 0; j <num-1-i; j++) {
49.                 if (contents.get(j).compareTo(contents.get(
   j+1))==1) {
50.                     Item item = contents.get(j);
51.                     contents.set(j, contents.get(j + 1));
52.                     contents.set(j + 1, item);
53.                 }
54.             }
55.         }
56.     }//用冒泡排序法将商品按价格从小到大进行排序
57.     public String getItemName(int a){
58.         return contents.get(a).getName();
59.     }//返回动态数组中第 a 项商品的名字
60.     public String toString() {
61.         return (this.getBuyer()+"的购物车中有
   "+this.getNum()+"件商品，共"+this.getTotalPrice()+"元。");
62.         //示例:张三的购物车中有 5 件商品，共 384 元。
63.     }
64. }
65.
66.
67.
```

Item.java:

```
8. package Practice3forood;
```

```
9.
10.public abstract class Item implements Comparable<Item>/*实现了部分 Comparable 接口，Item 为抽象类*/{
11.     private String name;
12.     private boolean supprtrefund=true;//可支持退货
13.     private double price;
14.     private int timeToReceive;//到货时间
15.     public Item(){};
16.     public Item(String name,boolean issupprt,double price,int time){
17.         this.name=name;
18.         supprtrefund=issupprt;
19.         this.price=price;
20.         timeToReceive=time;
21.     }
22.     public String getName(){
23.         return name;
24.     }
25.     public boolean getCanRe(){
26.         return supprtrefund;
27.     }
28.     public double getPrice(){
29.         return price;
30.     }
31.     public int getTime(){
32.         return timeToReceive;
33.     }
34.     public int compareTo(Item x){
35.         if(price>=x.getPrice())
36.             return 1;
37.         else return 0;
38.     }//实现 compareTo 函数，如果前者价格小于后者，返回0；否则返回1
39.}
40.
```

Electronicitem.java:

```
1. package Practice3forood;
2. public class Electronicitem extends Item{
3.     private int timeInService;//保修年限
4.     private boolean supprtrefund=true;//电子产品均可退货
5.     private String band;//品牌
6.     private String type;//种类
7.     public Electronicitem(String band,String type,int timeInService,String name, double price, int time) {
```

```
8.          super(name, true, price, time);
9.          this.band=band;
10.         this.type=type;
11.         this.timeInService=timeInService;
12.     }
13.     public String getBand(){
14.         return band;
15.     }
16.     public String getType(){
17.         return type;
18.     }
19.     public int getTimeInService(){
20.         return timeInService;
21.     }
22.     public String toString(){
23.         return (this.getBand()+"品牌的"+this.getType()+"价格为"+this.getPrice()+"，保修期为"+this.getTimeInService()+"年");
24.         //示例：HUAWEI 品牌的 phone 价格为 8848，保修期为 2 年
25.     }
26. }
27.
28.
```

Cloth.java:

```
1. package Practice3forood;
2.
3. public class Cloth extends Item{
4.     private String fabric;//布料
5.     private String band;//品牌
6.     public Cloth(String band,String fabric,String name,boolean issupprt,double price,int time){
7.         super(name, issupprt, price, time);
8.         this.fabric=fabric;
9.         this.band=band;
10.    }
11.    public String getBand(){
12.        return band;
13.    }
14.    public String getFabric(){
15.        return fabric;
16.    }
17.    public String toString(){
18.        return (this.getBand()+"品牌的"+this.getName()+"为"+this.getFabric()+"材质，价格为"+this.getPrice()+"元");
```

```
        //示例：LV 品牌的大衣为皮革材质，价格为12000 元
    }
}
```

Food.java:

```
package Practice3forood;

public class Food extends Item{
    private int expiredate; //保质期
    private boolean supprtrefund=false;//食品不可退货
    double weight=0.0;
    public Food(String name,double price, int time,int expiredate,double weight) {
        super(name, true, price, time);
        this.expiredate=expiredate;
        this.weight=weight;
    }
    public int getExpiredate(){
        return expiredate;
    }
    public double getWeight(){
        return weight;
    }
    public String toString(){
        return (this.getName()+"重"+this.getWeight()+"克,保质期有"+this.getExpiredate()+"天");
        //示例：薯片重 500 克，保质期有 180 天
    }
}
```

ShoppingCartTest.java;

```
package Practice3forood;

public class ShoppingCartTest {
    public static void main(String[] args){
        ShoppingCart myshcart=new ShoppingCart("Holly",0,0.0);
        Item elecitem=new Electronicitem("Apple","pad",2,"11proMax",8888,1);
        Item fooditem=new Food("juice",3.5,3,180,400);
        Item clothitem=new Cloth("ANTA","leather","coat",true,588,5);
        myshcart.addItem(elecitem);
        myshcart.addItem(fooditem);
        myshcart.addItem(clothitem);
```

```
12.         myshcart.sortByPrice();
13.         System.out.printf("商品按照价格由低到高为：");
14.         for(int i=0;i<myshcart.getNum();i++){
15.             System.out.println(myshcart.getItemName(i));
16.         }
17.         System.out.println(myshcart.toString());
18.         myshcart.deleteItem(fooditem);
19.         System.out.println("其中"+fooditem);
20.         System.out.printf("删除食品类后：");
21.         System.out.println(myshcart.toString());
22.         System.out.println("其中"+elecitem);
23.         System.out.println(clothitem+" ");
24.     }
25.}
```