# 实验6：抽象类和接口

## 实验目的和要求

掌握抽象类得设计和使用

掌握接口得设计及实现接口的类

掌握抽象的Number类的使用方法

掌握Comparable和Clone的接口的使用方法

理解抽象类和接口的异同

## 实验题目

### P455 13.1

原题：（三角形）设计一个扩展自抽象类GeometricObject的新的Triangle类。绘制Triangle类和GeometricObject类的UML图并实现Triangle类。编写一个测试程序，提示用户输入三角形的三条边、一种颜色以及一个表明该三角形是否填充的布尔值。程序应该根据用户的输入，使用这些边以及颜色和是否填充的信息，创建一个Triangle对象。程序应该显示面积、周长、颜色以及真或假来表明是否被填充。

源代码：package project;

import java.util.Date;

public abstract class GeometricObject {

private String color = "white";

private boolean filled;

private Date dateCreated;

protected GeometricObject() {

dateCreated = new Date();

}

protected GeometricObject(String color, boolean filled) {

dateCreated = new Date();

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public boolean isFilled() {

return filled;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public Date getDateCreated() {

return dateCreated;

}

@Override

public String toString() {

return "created on"+dateCreated+"\ncolor: "+color+" and filled: "+filled;

}

public abstract double getArea();

public abstract double getPerimeter();

}

class Triangle extends GeometricObject

{

private double a;

private double b;

private double c;

Triangle(double a, double b, double c,String color,boolean filled) {

super(color,filled);

if(a+b>c&&a+c>b&&c+b>a)

{

this.a = a;

this.b = b;

this.c = c;

}

}

@Override

public double getArea() {

double s;

double p;

p = (a+b+c)/2;

s = Math.sqrt((p-a)\*(p-b)\*(p-c));

return s;

}

@Override

public double getPerimeter() {

double l;

l = a + b + c;

return l;

}

@Override

public String toString() {

// TODO Auto-generated method stub

return super.toString()+"\n三边为："+a+","+b+","+c+"\n三角形周长和面积分别为："+this.getPerimeter()+"和"+this.getArea();

}

}

public class Test0 {

public static void main(String[] args) {

Triangle t = new Triangle(3,4,5,"red",true);

System.out.println(t.toString());

}

}

结果及截图：



### P455 13.5

原题：（将GeometricObject类变成可比较的）修改GeometricObject类以实现Comparable接口，并且在GeometricObject类中定义一个静态的求两个GeometricObject对象中较大者的max方法。画出UML图并实现这个新的GeometricObject类。编写一个测试程序，使用max方法求两个圆中的较大者和两个矩形中的较大者。

源代码：package project1;

import java.util.Date;

public abstract class GeometricObject implements Comparable<GeometricObject> {

private String color = "white";

private boolean filled;

private Date dateCreated;

protected GeometricObject() {

dateCreated = new Date();

}

protected GeometricObject(String color, boolean filled) {

dateCreated = new Date();

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public boolean isFilled() {

return filled;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public Date getDateCreated() {

return dateCreated;

}

@Override

public String toString() {

return "created on"+dateCreated+"\ncolor: "+color+" and filled: "+filled;

}

public abstract double getArea();

public abstract double getPerimeter();

public static int max(GeometricObject g1,GeometricObject g2) {

if(g1.getArea()>g2.getArea())

return 1;

else if(g1.getArea()==g2.getArea())

return 0;

else

return -1;

}

@Override

public int compareTo(GeometricObject g) {

if(this.getArea()>g.getArea())

return 1;

else if(this.getArea()==g.getArea())

return 0;

else

return -1;

}

}

public class Triangle extends GeometricObject{

private double a;

private double b;

private double c;

public Triangle(double a, double b, double c) {

super();

this.a = a;

this.b = b;

this.c = c;

}

@Override

public double getArea() {

double s;

double p;

p = (a+b+c)/2;

s = Math.sqrt((p-a)\*(p-b)\*(p-c));

return s;

}

@Override

public double getPerimeter() {

double l;

l = a + b + c;

return l;

}

}

public class Circle extends GeometricObject{

private double radius;

public Circle(double radius) {

super();

this.radius = radius;

}

@Override

public double getArea() {

double s;

s = Math.PI\*radius\*radius;

return s;

}

@Override

public double getPerimeter() {

double l;

l = 2\*Math.PI\*radius;

return l;

}

}

public class Test1 {

public static void main(String[] args) {

Circle c1 = new Circle(1);

Circle c2 = new Circle(2);

System.out.println(GeometricObject.max(c1, c2));

Triangle t1 = new Triangle(1,2,3);

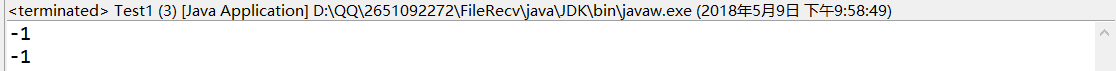
Triangle t2 = new Triangle(2,3,4);

System.out.println(GeometricObject.max(t1, t2));

}

}

结果及截图：



### P456 13.10

原题：（将Circle类变成可比较的）改写程序清单13-3的Rectangle类，它继承字Geometric-Object类并实现Comparable接口。覆盖Object类中的equals方法。当两个Rectangle对象面积相同时，则这两个对象是相同的。画出包括Rectangle、GeometricObject和Comparable的UML图。

源代码：

import java.util.Date;

public abstract class GeometricObject {

private String color = "white";

private boolean filled;

private Date dateCreated;

protected GeometricObject() {

dateCreated = new Date();

}

protected GeometricObject(String color, boolean filled) {

dateCreated = new Date();

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public boolean isFilled() {

return filled;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public Date getDateCreated() {

return dateCreated;

}

@Override

public String toString() {

return "created on"+dateCreated+"\ncolor: "+color+" and filled: "+filled;

}

public abstract double getArea();

public abstract double getPerimeter();

}

public class Rectangle extends GeometricObject{

private double width;

private double height;

public Rectangle() {}

public Rectangle(double width, double height) {

this.width = width;

this.height = height;

}

public Rectangle(double width, double height,String color,boolean filled) {

super(color,filled);

this.width = width;

this.height = height;

}

public void setWidth(double width) {

this.width = width;

}

public void setHeight(double height) {

this.height = height;

}

public double getWidth() {

return width;

}

public double getHeight() {

return height;

}

@Override

public double getArea() {

return width \* height;

}

@Override

public double getPerimeter() {

return 2\*(width + height);

}

public boolean equals(Rectangle r) {

if(this.getArea()==r.getArea())

return true;

else

return false;

}

}

public class ComparableRectangle extends Rectangle implements Comparable<ComparableRectangle>{

public ComparableRectangle(double width, double height) {

super(width, height);

}

@Override

public int compareTo(ComparableRectangle o) {

if(this.getArea()>o.getArea())

return 1;

else if(getArea()<o.getArea())

return -1;

else

return 0;

}

}

### P456 13.13

原题：（使得Course类可复制）重写程序清单10-6的Course类，增加一个clone方法，执行students域上的深毒复制。

源代码：

public class Course implements Cloneable{

private String courseName;

private String[] students = new String[100];

private int numberOfStudents;

public Course(String courseName) {

this.courseName = courseName;

}

public void addStudent(String student)

{

students[numberOfStudents] = student;

numberOfStudents++;

}

public String[] getStudents() {

return students;

}

public int getNumberOfStudents() {

return numberOfStudents;

}

public String getCourseName() {

return courseName;

}

public void dropStudent(String student){}

@Override

public Object clone() throws CloneNotSupportedException {

Course courseClone = (Course)super.clone();

courseClone.students = (String[]) (students.clone());

return courseClone;

}

}

public class Test3 {

public static void main(String[] args) throws CloneNotSupportedException {

// TODO Auto-generated method stub

Course c1 = new Course("Math");

Course c2 = (Course)c1.clone();

System.out.println(c1==c2);

System.out.println(c1.getCourseName()==c2.getCourseName());

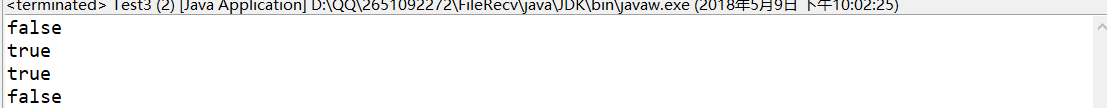
System.out.println(c1.getNumberOfStudents()==c2.getNumberOfStudents());

System.out.println(c1.getStudents()==c2.getStudents());

}

}

结果及截图：



### P457 13.17

原题：（数学：Complex类）一个复数是一个形式为a+bi的数，这里的a和b都是实数，i是根号下-1的平方根。数字a和b分别称为复数的实部和虚部。

设计一个名为Complex的复数来表示复数以及完成复数运算的add、substract、multiply、divide和abs方法，并且覆盖toString方法以返回一个表示复数的字符串。方法toString返回字符串a+bi。如果b是0，那么它只返回a。Complex类应该也实现Cloneable接口。

提三个构造方法Complex（a,b）、Complex（a）和Complex（）。Complex（）创建数字0的Complex对象，而Complex（a）创建一个b为0的对象。还提供getRealPart（）和getImaginaryPart方法以返回复数的实部和虚部。

编写一个测试程序，提示用户输入两个复数，然后显示它们做加、减、乘、除之后的结果。

源代码：

public class Complex implements Cloneable{

private double a;

private double b;

public Complex()

{

this.a = 0;

this.b = 0;

}

public Complex(double a)

{

this.a = a;

this.b = 0;

}

public Complex(double a,double b)

{

this.a = a;

this.b = b;

}

public double getRealPart()

{

return a;

}

public double getImaginaryPart()

{

return b;

}

public String add(Complex c)

{

double a;

double b;

a = this.a + c.a;

b = this.b + c.b;

return "("+this.a+" + "+this.b+"i)"+" + "+"("+c.a+" + "+c.b+"i)"+" = "+a+" + "+b+"i";

}

public String substract(Complex c)

{

double a;

double b;

a = this.a - c.a;

b = this.b - c.b;

return "("+this.a+" + "+this.b+"i)"+" - "+"("+c.a+" + "+c.b+"i)"+" = "+a+" + "+b+"i";

}

public String multiply(Complex c)

{

double a;

double b;

a = this.a\*c.a - this.b\*c.b;

b = this.b\*c.a + this.a\*c.b;

return "("+this.a+" + "+this.b+"i)"+" \* "+"("+c.a+" + "+c.b+"i)"+" = "+a+" + "+b+"i";

}

public String divide(Complex c)

{

double a;

double b;

a = (this.a\*c.a + this.b\*c.b)/(c.a\*c.a+c.b\*c.b);

b = (this.b\*c.a - this.a\*c.b)/(c.a\*c.a+c.b\*c.b);

return "("+this.a+" + "+this.b+"i)"+" / "+"("+c.a+" + "+c.b+"i)"+" = "+a+" + "+b+"i";

}

public String abs()

{

double absValue;

absValue = Math.sqrt(this.a\*this.a + this.b\*this.b);

return "|("+this.a+" + "+this.b+"i)|"+" = "+absValue;

}

public String toString()

{

if(b!=0)

return a+"+"+b+"i";

else

return a+"";

}

@Override

public Object clone() throws CloneNotSupportedException {

return super.clone();

}

}

import java.util.Scanner;

public class Test4 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner in = new Scanner(System.in);

System.out.print("Enter the first complex number:");

double a = in.nextDouble();

double b = in.nextDouble();

System.out.print("Enter the second complex number:");

double c = in.nextDouble();

double d = in.nextDouble();

Complex c1 = new Complex(a,b);

Complex c2 = new Complex(c,d);

System.out.println(c1.add(c2));

System.out.println(c1.substract(c2));

System.out.println(c1.multiply(c2));

System.out.println(c1.divide(c2));

System.out.println(c1.abs());

}

}

结果及截图：

