|  |  |
| --- | --- |
| **面向对象程序设计 作业报告** | |
| 第2次 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
|  | |
|  | |
|  | |
| **姓名** |  |
| **班级** |  |
| **学号** |  |
| **电话** |  |
| **Email** |  |
| **日期** |  |

# 目录

### 题目1 创建一个用来表示时间的类 1

1. 题目 1
2. 数据设计 2
3. 算法设计 2
4. 主干代码说明 2
5. 运行结果展示 4
6. 总结和收获 5

### 题目2 创建自己的GUI图形 5

1. 题目 5
2. 数据设计 6
3. 算法设计 6
4. 主干代码说明 8
5. 运行结果展示 9
6. 总结和收获 9

### 题目3 接口的威力 10

1. 题目 10
2. 数据设计 10
3. 算法设计 11
4. 主干代码说明 11
5. 运行结果展示 12
6. 总结和收获 12

### 题目4 撰写继承、多态和接口方面的知识梳理 12

1. 题目 12
2. 知识梳理 13

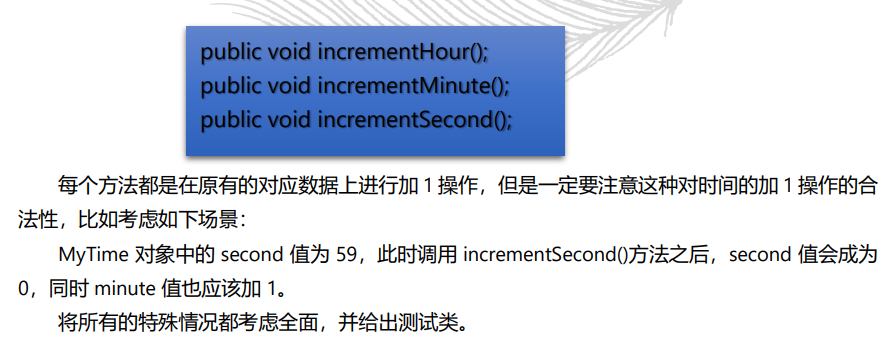
**附录：各个题的源代码 13**

# 题目1 创建一个用来表示时间的类

##### 题目







（题目中所提供程序附在附录中）

##### 数据设计

MyTime类用于存储时间数据，涉及的数据有：时、分、秒；分别定义3个int类型的变量进行存储即可。

##### 算法设计

1. toUniversalString()方法：根据题目要求，分别使用3个if语句判断时、分、秒是否合法，若不合法输出要求语句；若都合法，使用String.format()方法返回符合要求的语句。
2. toString()方法：需要考虑AM/PM的输出。时、分、秒合法的情况下，若时小于12，返回语句最后为AM；若时为12，返回语句最后为PM；若时大于12，返回语句中时减去12且为PM。
3. incrementHour()方法：将时加一；若此时时等于24，将之赋值为0。
4. incrementMinute()方法：将分加一；若此时分等于60，将之赋值为0，调用incrementHour()方法将时加一。
5. incrementSecond()方法：将秒加一；若此时秒等于60，将之赋值为0，调用incrementMinute()方法将分加一。

##### 主干代码说明

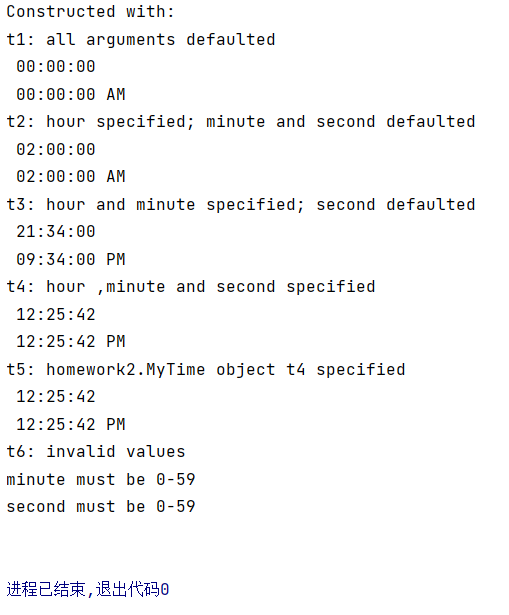
主干代码如下：

1. public String toUniversalString(){
2. if(hour>23){
3. System.out.println("hour must be 0-23");
4. }
5. if(minute>59){
6. System.out.println("minute must be 0-59");
7. }
8. if(second>59){
9. System.out.println("second must be 0-59");
10. }
11. if(hour<24&&minute<60&&second<60){
12. return String.format("%02d:%02d:%02d",hour,minute,second);
13. }
14. else{
15. return "";
16. }
17. }
18. public String toString(){
19. if(hour<24&&minute<60&&second<60){
20. if(hour<12){*//a.m.*
21. return String.format("%02d:%02d:%02d AM",hour,minute,second);
22. }
23. else if(hour>12){*//p.m.*
24. return String.format("%02d:%02d:%02d PM",hour-12,minute,second);
25. }
26. else if(hour==12){*//moon*
27. return String.format("%02d:%02d:%02d PM",hour,minute,second);
28. }
29. }
30. return "";
31. }
32. public void incrementHour(){
33. hour++;
34. if(hour==24){
35. hour=0;
36. }
37. }
38. public void incrementMinute(){
39. minute++;
40. if(minute==60){
41. minute=0;
42. incrementHour();
43. }
44. }
45. public void incrementSecond(){
46. second++;
47. if(second==60){
48. second=0;
49. incrementMinute();
50. }
51. }

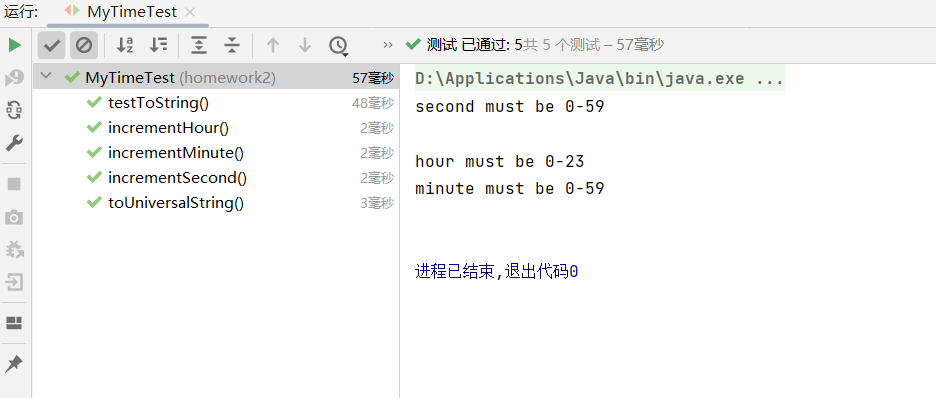
以上代码实现了算法设计中所述的方法。代码中形如"%02d:%02d:%02d AM"保证了时、分、秒的输出均为两位数字；时、分、秒加一的方法中考虑到了进位的情况。

##### 运行结果展示

TestTime.java的运行结果：



测试类MyTimeTest.java的运行结果：

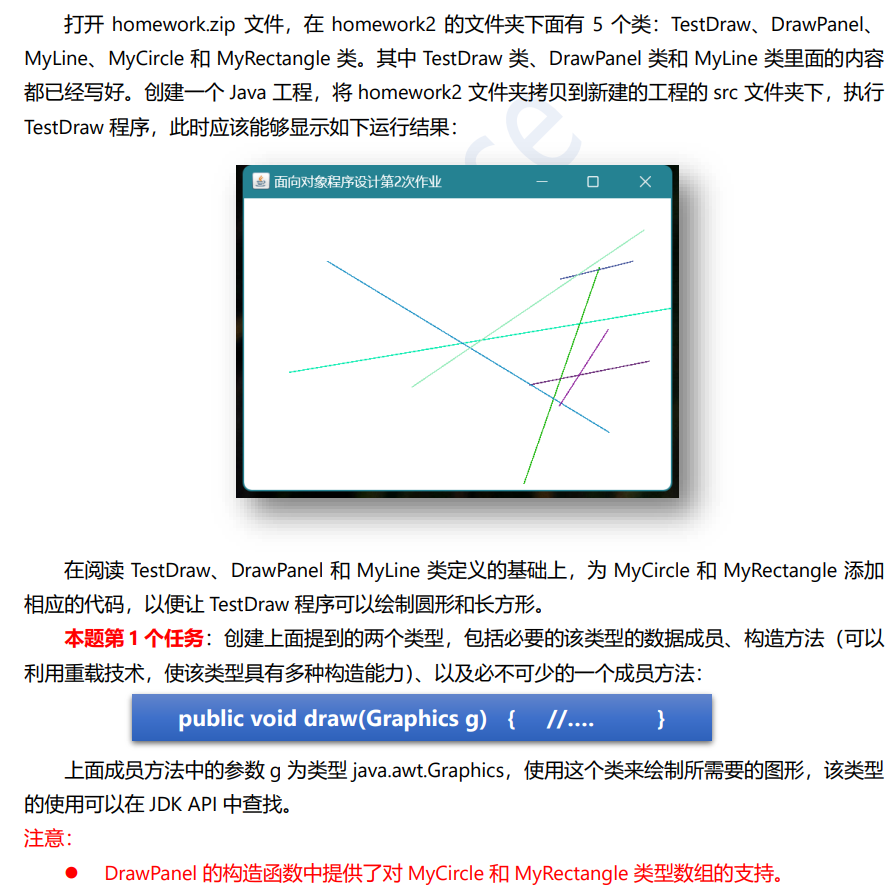


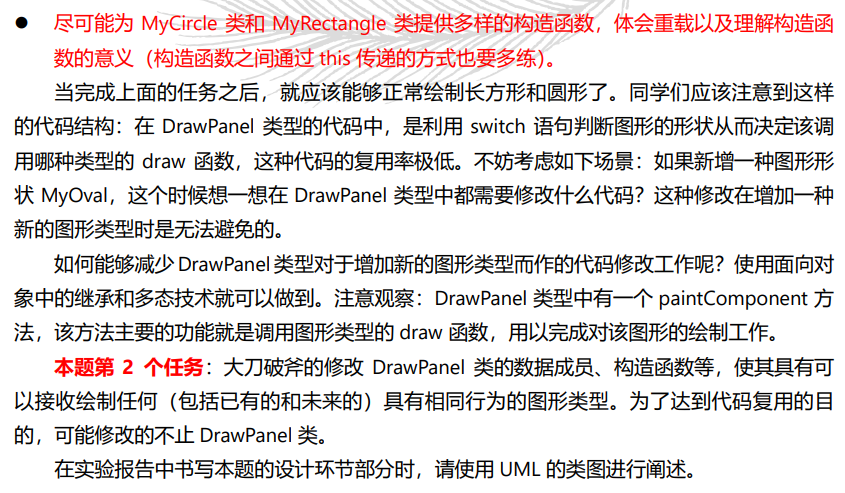
##### 总结和收获

本题练习了设计类和方法、实现类的构造器、类变量的赋值、方法的实现等，同时也练习了测试类的编写、类的测试方法过程。

# 题目2 创建自己的GUI图形

##### 题目





##### 数据设计

任务一：

1. MyCircle类被用来表示圆形，需要的数据成员有：圆心的坐标、圆的半径、圆的颜色，可分别使用int x，int y，int r，Color color来表示。
2. MyRectangle类被用来表示矩形，需要的数据成员有：矩形的坐标（取左上角）、矩形的长、矩形的宽、矩形的颜色，可分别使用int x，int y，int width， int height，Color color来表示。

任务二：

1. 为了使DrawPanel类可以处理任何具有相同行为的图形类型，需要将类中包含的多种图形类型统一。新定义一个超类MyGraph，DrawPanel类中只需包含一个MyGraph[] graphs变量，而无需含有多种图形类型的变量。
2. 超类MyGraph中无需含有数据，确定每个图形的数据可以由子类自行定义。

##### 算法设计

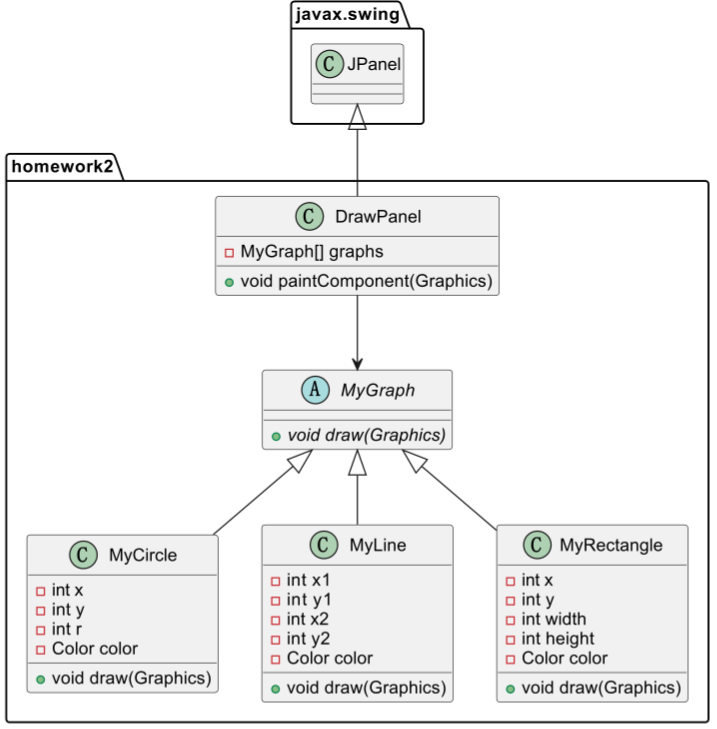
任务一·构造函数：

1. 为MyCircle类提供4种构造方法，分别是参数齐全的构造方法、不提供坐标的构造方法、不提供颜色的构造方法、不提供颜色和坐标的构造方法；
2. 为MyRectangle类提供7种构造方法，分别是不提供参数的构造方法、矩形构造方法、不提供坐标的矩形构造方法、不提供颜色的矩形构造方法、不提供颜色和坐标的矩形构造方法、正方形构造方法、不提供坐标的正方形构造方法。

任务一·draw方法

1. 仿照MyLine中的draw方法，先调用java.awt.Graphics中的setColor方法设定颜色，再调用其中的drawOval或drawRect方法绘制出圆形或矩形。

任务二·UML图



##### 主干代码说明

MyGraph.java

1. package homework2;
2. import java.awt.Graphics;
3. public abstract class MyGraph {
4. public abstract void draw(Graphics g);
5. }

定义了一个名为MyGraph的抽象类，提供一个draw(Grapics g)方法，这样就可以在DrawPanel中定义MyGraph类型的变量，并直接调用MyGraph.draw(Grapics g)方法。具体的draw(Grapics g)将由子类定义。

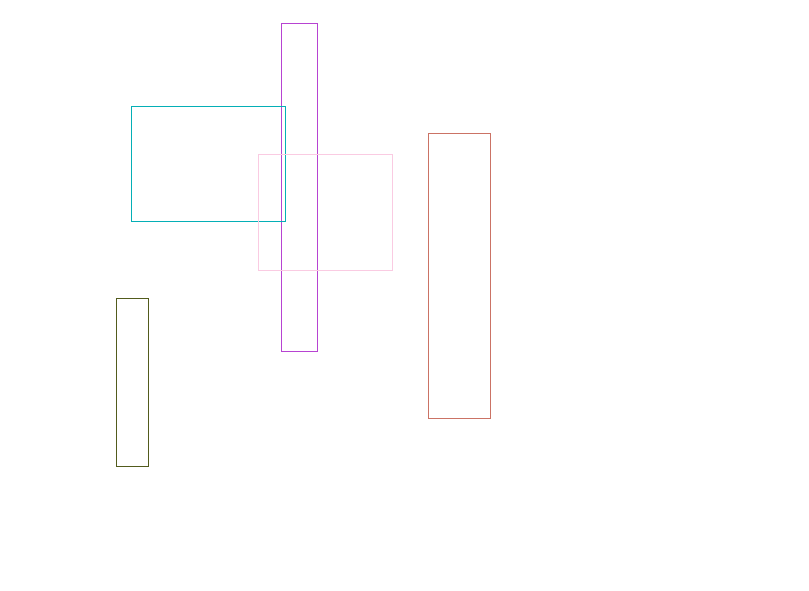
1. public DrawPanel(MyGraph[] graphs){
2. setBackground(Color.WHITE);
3. if (graphs == null)
4. {···}
5. if (!graphs.getClass().isArray())
6. {···}
7. this.graphs = graphs;
8. type = GRAPH;
9. }
10. public void paintComponent(Graphics g){
11. super.paintComponent(g);
12. switch(type){
13. case GRAPH:
14. for(MyGraph graph : graphs)
15. graph.draw(g);
16. break;
17. case NULL:
18. ···
19. case NOTARRAY:
20. ···
21. }
22. }

以上是对DrawPanel类的构造函数和paintComponent方法的修改。此时只需要初始化一个MyGraph对象、调用MyGraph的draw方法。

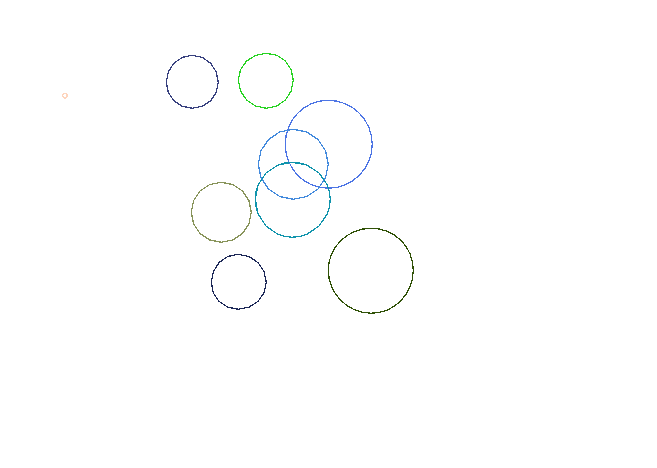
##### 运行结果展示

DrawTest.java进行测试的运行结果：

画矩形：



画圆：

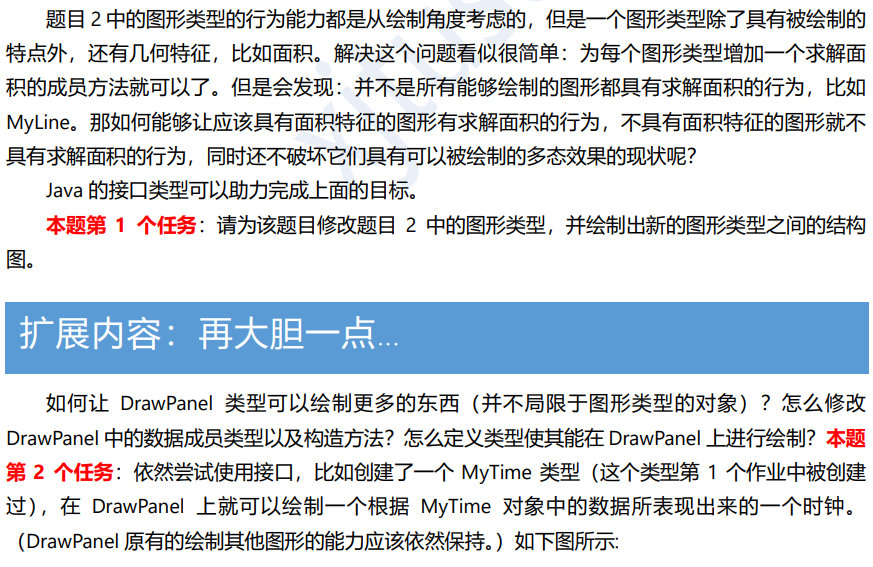


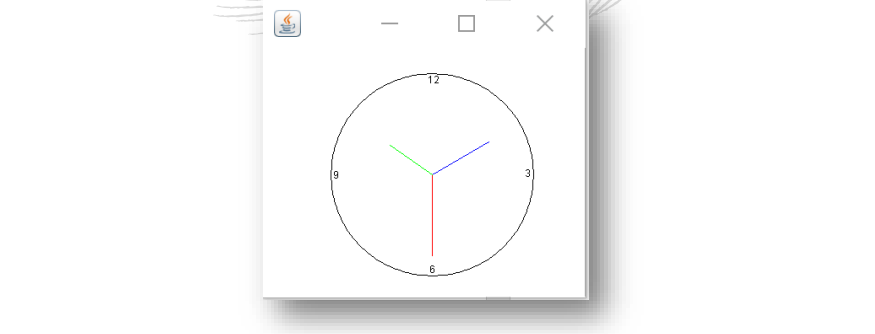
##### 总结和收获

本题了解了java.awt.Graphics的一些方法，练习了使用类的继承、多态，体会通过继承和多态简化绘图方法。

# 题目3 接口的威力

##### 题目





##### 数据设计

方法一

根据题目要求，可以定义两个接口Drawable，AreaMeasurable，由此需将DrawPanel类中的数据改为Drawable[] drawables；

方法二

使MyTime类实现接口Drawable并实现方法draw即可，无需对数据进行更改。

##### 算法设计

方法一

使MyCircle，MyRectangle类实现Drawable，AreaMeasurable接口，并为他们增加getArea()方法的实现即可。

方法二

需在MyTime类中实现draw方法：先绘制一个圆作为表盘，在通过绘制字符串绘制出表上的数字，最后根据对象中的数据绘制表针。

##### 主干代码说明

为MyRectangle.java添加的方法

1. public double getArea()
2. {
3. return width\*height;
4. }

为MyCircle.java添加的方法

1. public double getArea()
2. {
3. return r\*r\*Math.PI;
4. }

Drawable.java

1. package homework2;
2. import java.awt.Graphics;
3. public interface Drawable {
4. void draw(Graphics g);
5. }

AreaMeasurable.java

1. package homework2;
2. public interface AreaMeasurable {
3. double getArea();
4. }

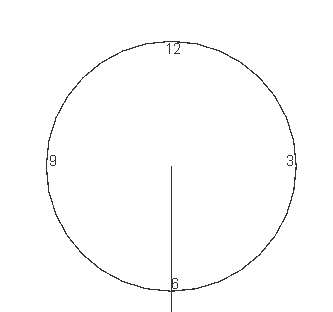
由以上代码实现了可以计算面积的图形拥有计算面积的方法，而不影响绘制的方法。

为MyTime.java添加的方法

1. public void draw(Graphics g){
2. g.drawOval(50,50,200,200);
3. g.drawString("12", 145, 60);
4. g.drawString("3", 242, 150);
5. g.drawString("6", 150, 248);
6. g.drawString("9", 52, 150);
7. g.drawLine(150, 150, (int)(150+50\*Math.sin(Math.abs(hour-12)/12)), (int)(150+50\*Math.cos(Math.abs(hour-12)/12)));
8. g.drawLine(150, 150, (int)(150+100\*Math.sin(Math.abs(minute)/60)), (int)(150+100\*Math.cos(Math.abs(minute)/60)));
9. g.drawLine(150, 150, (int)(150+140\*Math.sin(Math.abs(second)/60)), (int)(150+140\*Math.cos(Math.abs(second)/60)));
10. }

以上实现了绘制时钟。

##### 运行结果展示

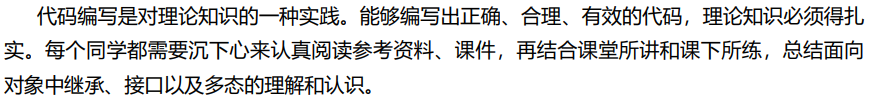


##### 总结和收获

本题练习了接口的定义和使用、以及如何通过实现接口来实现一个类的多种属性。

# 题目4 撰写继承、多态和接口方面的知识梳理

##### 题目



##### 知识梳理

继承是指一个类获得一个超类的所有非私有属性和方法、使之与超类拥有相同的行为。通过继承，子类可以直接获得超类的方法，亦可以对超类的方法进行重写。但是，一个类只能继承一个超类。

接口是抽象方法的集合。类描述对象的属性、实现对象的方法，接口用来指定类要实现的方法。若一个非抽象类实现接口，它必须对接口中的方法进行实现。接口可以帮助类实现多继承。

多态是同一个行为具有多个不同表现形式或形态的能力。通过重写、接口或抽象类，同一个方法在不同的对象上会有不同的实现方法，这就是多态的体现。

# 附录 每个题的源代码

##### 题目1

MyTime.java

1. package homework2;
2. public class MyTime {
3. *//variable and initialization*
4. private int hour=0,minute=0,second=0;
5. *//constructor*
6. public MyTime(){}
7. public MyTime(int hour){
8. this.hour=hour;
9. }
10. public MyTime(int hour,int minute){
11. this.hour=hour;
12. this.minute=minute;
13. }
14. public MyTime(int hour,int minute,int second){
15. this.hour=hour;
16. this.minute=minute;
17. this.second=second;
18. }
19. public MyTime(MyTime time){
20. this.hour=time.hour;
21. this.minute=time.minute;
22. this.second=time.second;
23. }
24. public String toUniversalString(){
25. if(hour>23){
26. System.out.println("hour must be 0-23");
27. }
28. if(minute>59){
29. System.out.println("minute must be 0-59");
30. }
31. if(second>59){
32. System.out.println("second must be 0-59");
33. }
34. if(hour<24&&minute<60&&second<60){
35. return String.format("%02d:%02d:%02d",hour,minute,second);
36. }
37. else{
38. return "";
39. }
40. }
41. public String toString(){
42. if(hour<24&&minute<60&&second<60){
43. if(hour<12){*//a.m.*
44. return String.format("%02d:%02d:%02d AM",hour,minute,second);
45. }
46. else if(hour>12){*//p.m.*
47. return String.format("%02d:%02d:%02d PM",hour-12,minute,second);
48. }
49. else if(hour==12){*//moon*
50. return String.format("%02d:%02d:%02d PM",hour,minute,second);
51. }
52. }
53. return "";
54. }
55. public void incrementHour(){
56. hour++;
57. if(hour==24){
58. hour=0;
59. }
60. }
61. public void incrementMinute(){
62. minute++;
63. if(minute==60){
64. minute=0;
65. incrementHour();
66. }
67. }
68. public void incrementSecond(){
69. second++;
70. if(second==60){
71. minute=0;
72. incrementMinute();
73. }
74. }
75. }

TestTime.java

1. package homework2;
2. public class TestTime {
3. public static void main(String[] args) {
4. MyTime t1 = new MyTime();
5. MyTime t2 = new MyTime(2);
6. MyTime t3 = new MyTime(21,34);
7. MyTime t4 = new MyTime(12, 25, 42);
8. MyTime t5 = new MyTime(t4);
9. System.out.println("Constructed with:");
10. System.out.println("t1: all arguments defaulted");
11. System.out.printf(" %s\n", t1.toUniversalString());
12. System.out.printf(" %s\n", t1.toString());
13. System.out.println("t2: hour specified; minute and second defaulted");
14. System.out.printf(" %s\n", t2.toUniversalString());
15. System.out.printf(" %s\n", t2.toString());
16. System.out.println("t3: hour and minute specified; second defaulted");
17. System.out.printf(" %s\n", t3.toUniversalString());
18. System.out.printf(" %s\n", t3.toString());
19. System.out.println("t4: hour ,minute and second specified");
20. System.out.printf(" %s\n", t4.toUniversalString());
21. System.out.printf(" %s\n", t4.toString());
22. System.out.println("t5: homework2.MyTime object t4 specified");
23. System.out.printf(" %s\n", t5.toUniversalString());
24. System.out.printf(" %s\n", t5.toString());
25. *//when initialize t6 with invalid values,please output error information*
26. MyTime t6 = new MyTime(15, 74, 99);
27. System.out.println("t6: invalid values");
28. System.out.printf("%s\n", t6.toUniversalString());
29. }
30. }

测试类 MyTimeTest.java

1. package homework2;
2. import static org.junit.jupiter.api.Assertions.\*;
3. class MyTimeTest {
4. @org.junit.jupiter.api.Test
5. void toUniversalString() {
6. assertEquals("15:00:00",new MyTime(15).toUniversalString());
7. assertEquals("23:45:50",new MyTime(23,45,50).toUniversalString());
8. System.out.println(new MyTime(11,45,66).toUniversalString());
9. System.out.println(new MyTime(25,70,0).toUniversalString());
10. }
11. @org.junit.jupiter.api.Test
12. void testToString() {
13. assertEquals("03:25:00 PM",new MyTime(15,25).toString());
14. assertEquals("12:10:45 PM",new MyTime(12,10,45).toString());
15. assertEquals("07:00:00 AM",new MyTime(7).toString());
16. }
17. @org.junit.jupiter.api.Test
18. void incrementHour() {
19. MyTime test1=new MyTime(23,45,50);
20. test1.incrementHour();
21. assertEquals("00:45:50 AM",test1.toString());
22. MyTime test2=new MyTime(14,11);
23. test2.incrementHour();
24. assertEquals("15:11:00",test2.toUniversalString());
25. }
26. @org.junit.jupiter.api.Test
27. void incrementMinute() {
28. MyTime test1=new MyTime(23,59,59);
29. test1.incrementMinute();
30. assertEquals("00:00:59 AM",test1.toString());
31. MyTime test2=new MyTime(19,1);
32. test2.incrementMinute();
33. assertEquals("19:02:00",test2.toUniversalString());
34. MyTime test3=new MyTime(15,59);
35. test3.incrementMinute();
36. assertEquals("04:00:00 PM",test3.toString());
37. }
38. @org.junit.jupiter.api.Test
39. void incrementSecond() {
40. MyTime test1=new MyTime(23,59,59);
41. test1.incrementSecond();
42. assertEquals("00:00:00 AM",test1.toString());
43. MyTime test2=new MyTime(11,45,14);
44. test2.incrementSecond();
45. assertEquals("11:45:15",test2.toUniversalString());
46. MyTime test3=new MyTime(15,59,59);
47. test3.incrementSecond();
48. assertEquals("04:00:00 PM",test3.toString());
49. }
50. }

##### 题目2

MyRectangle.java

1. package homework2;
2. import java.awt.Graphics;
3. import java.awt.Color;
4. public class MyRectangle extends MyGraph {
5. private int x = 0;
6. private int y = 0;
7. private int width = 1;
8. private int height = 1;
9. private Color color = Color.black;
10. public MyRectangle(){}
11. public MyRectangle(int x,int y,int width,int height,Color color){*//Ordinary constructor*
12. this.x=x;
13. this.y=y;
14. this.width=width;
15. this.height=height;
16. this.color=color;
17. }
18. public MyRectangle(int x,int y,int width,int height){*//Ordinary constructor without color*
19. this.x=x;
20. this.y=y;
21. this.width=width;
22. this.height=height;
23. }
24. public MyRectangle(int width,int height,Color color){*//Ordinary constructor without coordinates*
25. this.width=width;
26. this.height=height;
27. this.color=color;
28. }
29. public MyRectangle(int width,int height){*//Ordinary constructor without coordinates and color*
30. this.width=width;
31. this.height=height;
32. }
33. public MyRectangle(int x,int y,int sideLength,Color color){*//Square constructor*
34. this.x=x;
35. this.y=y;
36. this.width=sideLength;
37. this.height=sideLength;
38. this.color=color;
39. }
40. public MyRectangle(int sideLength,Color color){*//Square constructor without coordinates*
41. this.width=sideLength;
42. this.height=sideLength;
43. this.color=color;
44. }
45. public void draw(Graphics g)
46. {
47. g.setColor(color);
48. g.drawRect(x,y,width,height);
49. }
50. }

MyCircle.java

1. package homework2;
2. import java.awt.Graphics;
3. import java.awt.Color;
4. public class MyCircle extends MyGraph {
5. private int x;
6. private int y;
7. private int r;
8. private Color color = Color.black;
9. public MyCircle(int x, int y, int r, Color color) {*//Ordinary constructor*
10. this.x = x;
11. this.y = y;
12. this.r = r;
13. this.color = color;
14. }
15. public MyCircle(int r, Color color) {*//Without coordinates*
16. this.x = r;
17. this.y = r;
18. this.r = r;
19. this.color = color;
20. }
21. public MyCircle(int x, int y, int r) {*//Without color*
22. this.x = x;
23. this.y = y;
24. this.r = r;
25. }
26. public MyCircle(int r) {*//Without coordinates and color*
27. this.x = r;
28. this.y = r;
29. this.r = r;
30. }
31. public void draw(Graphics g)
32. {
33. g.setColor(color);
34. g.drawOval(x-r,y-r,2\*r,2\*r);
35. }
36. }

MyGraph.java

1. package homework2;
2. import java.awt.Graphics;
3. public abstract class MyGraph {
4. public abstract void draw(Graphics g);
5. }

DrawPanel.java

1. package homework2;
2. import javax.swing.JPanel;
3. import java.awt.Color;
4. import java.awt.Graphics;
5. public class DrawPanel extends JPanel {
6. private static final long serialVersionUID = 1L;
7. private MyGraph[] graphs;
8. private int type;
10. private static final int NONE = 0;
11. private static final int GRAPH = 1;
12. private static final int NULL = 2;
13. private static final int NOTARRAY = 3;
15. public DrawPanel()
16. {
17. setBackground(Color.BLACK);
18. type = NONE;
19. }
20. public DrawPanel(MyGraph[] graphs){
21. setBackground(Color.WHITE);
22. if (graphs == null)
23. {
24. type = NULL;
25. return;
26. }
27. if (!graphs.getClass().isArray())
28. {
29. type = NOTARRAY;
30. return;
31. }
32. this.graphs = graphs;
33. type = GRAPH;
34. }
36. public void paintComponent(Graphics g)
37. {
38. super.paintComponent(g);
39. switch(type){
40. case GRAPH:
41. for(MyGraph graph : graphs)
42. graph.draw(g);
43. break;
44. case NULL:
45. g.drawString("在DrawPanel的构造函数中，你传递的引用参数是NULL。", 50, 50);
46. break;
47. case NOTARRAY:
48. g.drawString("在DrawPanle的构造函数中，你传递的引用参数必须是某个形状的数组类型。", 50, 50);
49. break;
50. }
51. }
52. }

##### 题目3

Drawable.java

1. package homework2;
2. import java.awt.Graphics;
3. public interface Drawable {
4. void draw(Graphics g);
5. }

AreaMeasurable.java

1. package homework2;
2. public interface AreaMeasurable {
3. double getArea();
4. }

DrawPanel.java

1. package homework2;
2. import javax.swing.JPanel;
3. import java.awt.Color;
4. import java.awt.Graphics;
5. public class DrawPanel extends JPanel {
6. private static final long serialVersionUID = 1L;
7. private Drawable[] drawables;
8. private int type;
10. private static final int NONE = 0;
11. private static final int GRAPH = 1;
12. private static final int NULL = 2;
13. private static final int NOTARRAY = 3;
15. public DrawPanel()
16. {
17. setBackground(Color.BLACK);
18. type = NONE;
19. }
20. public DrawPanel(Drawable[] drawables){
21. setBackground(Color.WHITE);
22. if (drawables == null)
23. {
24. type = NULL;
25. return;
26. }
27. if (!drawables.getClass().isArray())
28. {
29. type = NOTARRAY;
30. return;
31. }
32. this.drawables = drawables;
33. type = GRAPH;
34. }
36. public void paintComponent(Graphics g)
37. {
38. super.paintComponent(g);
39. switch(type){
40. case GRAPH:
41. for(Drawable drawable : drawables)
42. drawable.draw(g);
43. break;
44. case NULL:
45. g.drawString("在DrawPanel的构造函数中，你传递的引用参数是NULL。", 50, 50);
46. break;
47. case NOTARRAY:
48. g.drawString("在DrawPanle的构造函数中，你传递的引用参数必须是某个形状的数组类型。", 50, 50);
49. break;
50. }
51. }
52. }

为MyRectangle.java添加的方法

1. public double getArea()
2. {
3. return width\*height;
4. }

为MyCircle.java添加的方法

1. public double getArea()
2. {
3. return r\*r\*Math.PI;
4. }

为MyTime.java添加的方法