目录

题目 1：创建一个用来表示时间的类

1.1 题目

1. 创建一个用来存储时间数据的MyTime类类型，当创建完这个类以后，应该保证下面的测试程序可以运行，且运行的结果如图所示。

Listing 1: src/hw2/p1/TestTime.java, L5-L31

```
1         // Official Test
2         MyTime t1 = new MyTime();
3         MyTime t2 = new MyTime(2);
4         MyTime t3 = new MyTime(21, 34);
5         MyTime t4 = new MyTime(12, 25, 42);
6         MyTime t5 = new MyTime(t4);
7
8         System.out.println("Constructed with:");
9         System.out.println("t1: all arguments defaulted");
10        System.out.printf(" %s\n", t1.toUniversalString());
11        System.out.printf(" %s\n", t1.toString());
12        System.out.println("t2: hour specified; minute and second
              defaulted");
13        System.out.printf(" %s\n", t2.toUniversalString());
14        System.out.printf(" %s\n", t2.toString());
15        System.out.println("t3: hour and minute specified; second
              defaulted");
16        System.out.printf(" %s\n", t3.toUniversalString());
17        System.out.printf(" %s\n", t3.toString());
18        System.out.println("t4: hour, minute and second specified");
19        System.out.printf(" %s\n", t4.toUniversalString());
20        System.out.printf(" %s\n", t4.toString());
21        System.out.println("t5: MyTime object t4 specified");
22        System.out.printf(" %s\n", t5.toUniversalString());
23        System.out.printf(" %s\n", t5.toString());
24        // when initialize t6 with invalid values,please output error
              information
```

```
25          MyTime t6 = new MyTime(15, 74, 99);
26          System.out.println("t6: invalid values");
27          System.out.printf("%s\n", t6.toUniversalString());
```

```
Constructed with:
t1: all arguments defaulted
    00:00:00
    12:00:00 AM
t2: hour specified; minute and
second defaulted
    02:00:00
    02:00:00 AM
t3: hour and minute specified;
second defaulted
    21:34:00
    09:34:00 PM
t4: hour ,minute and second
specified
    12:25:42
    12:25:42 PM
t5: MyTime object t4 specified
    12:25:42
    12:25:42 PM
t6: invalid values
minute must be 0-59
second must be 0-59
```

图 1: 运行结果样例

2. 为MyTime类增加三个成员方法：

```
1     public void incrementHour();
2     public void incrementMinute();
3     public void incrementSecond();
```

每个方法都是在原有的对应数据上进行加 1 操作，但是一定要注意这种对时间的加 1 操作的合法性，比如考虑如下场景：MyTime对象中的second值为 59，此时调用incrementSecond()方法之后，second值会成为 0，同时minute值也应该加 1。将所有的特殊情况都考虑全面，并给出测试类。

1.2 题目分析

本题需要对着已有的测试设计编写MyTime类，并对increment系列函数编写测试。

- **Field**. 参照代码注释可知，可以定义三个int变量hour ,minute,second分别代表小时、分钟、秒钟，24 小时制。

- **Constructor Method**. 测试代码中一共出现了 5 种构造方法
  - `MyTime()`,`MyTime(int)`,`MyTime(int, int)`, `MyTime(int, int, int)`; 这 4 种构造方法都是直接将参数赋给类的变量，分别为默认、只赋时钟、只赋分钟、全赋值，未赋值的变量默认值为 0；
  - `MyTime(MyTime)`这一构造方法是将参量复制给新的对象，即复制参量中的三个变量给新的对象。
  - 值得注意的是，除了默认的构造方法外，构造方法均可以通过调用其他的构造方法实现，这样可以极大减少代码量，提高代码复用性。
- **Other Method**. 本题目共要求实现编写 5 个实例化方法
  - `String toUniversalString()`. 本方法需要先判断`this`数据是否合法，根据判断结果返回相应的字符串。
    1. 判断`this`的数据是否合法，因为错误信息需要检查所有的数据，所以设置一个初值为 `true`的`boolean`变量`flag`，在判断过程中任一的判断错误都要将`flag`置为`false`.
    2. 如果不合法，输出判断过程中生成的错误信息
    3. 如果合法，根据`this`的数据输出时间信息。此处需要注意字符串的输出格式：前导空格和 Java format 中的`%02d`.
  - `String toString()`. 本方法只需返回 12 小时制的时间字符串，所以只需注意`hour`数据的处理和格式问题。
    1. 参看测试代码可知，如果`hour%12`为 0，我们需要输出为 12 点，其他情况取余即可
    2. 格式问题可以参照上一个方法中的第三条，并在最后附上`AM/PM`
  - `increment`系列方法. 这几个方法逻辑基本类似，都是判断数据`++`后是否超过上界，如果越界了需要将该数据置为 0，并将更高级的单位数据`++`。
    - 基于这种`++`操作的传递性,我们可以通过调用上一级单位的`increment`方法简化我们的代码逻辑。
- **Test for Increment**. 测试代码只需确定每个`increment`函数都正确无误，并测试其易错的进位即可，根据这个思路设计 `01:02:03`和`23:59:59`. 两个数据。

1.3 结果展示

```
Constructed with:
t1: all arguments defaulted
    00:00:00
    12:00:00 AM
t2: hour specified; minute and second defaulted
    02:00:00
    02:00:00 AM
t3: hour and minute specified; second defaulted
    21:34:00
    09:34:00 PM
t4: hour, minute and second specified
    12:25:42
    12:25:42 PM
t5: MyTime object t4 specified
    12:25:42
    12:25:42 PM
t6: invalid values
minute must be 0-59
second must be 0-59

-----NEW TESTS FOR INCREMENT-----
t7 is initialized as    01:02:03
  after t7.incrementHour(), t7 is    02:02:03
  after t7.incrementMinute(), t7 is    02:03:03
  after t7.incrementSecond(), t7 is    02:03:04
t8 is initialized as    23:59:59
  after t8.incrementSecond(), t8 is    00:00:00
```

图 2: 运行结果

1.4 主干代码

Listing 2: MyTime.java L8-L87

```
1    public class MyTime {
2    int hour;
3    int minute;
4    int second;
5
```

```
6       MyTime() {
7           hour = 0;
8           minute = 0;
9           second = 0;
10      }
11
12      MyTime(int h) {
13          this();
14          hour = h;
15      }
16
17      public MyTime(int h, int m) {
18          this(h);
19          minute = m;
20      }
21
22      public MyTime(int h, int m, int s) {
23          this(h, m);
24          second = s;
25      }
26
27      public MyTime(MyTime t4) {
28          this(t4.hour, t4.minute, t4.second);
29      }
30
31      boolean ValidJudge(int num) {
32          return num < 0 || num >= 60;
33      }
34
35      public String toUniversalString() {
36          boolean flag = true;
37          String s = "";
38
39          if (ValidJudge(hour)) {
40              flag = false;
41              s += "hour must be 0-23\n";
```

```
42          }
43          if (ValidJudge(minute)) {
44              flag = false;
45              s += "minute must be 0-59\n";
46          }
47          if (ValidJudge(second)) {
48              flag = false;
49              s += "second must be 0-59\n";
50          }
51          if (flag) return String.format("   %02d:%02d:%02d", hour, minute,
                second);
52          else return s;
53      }
54
55      @Override
56      public String toString() {
57          String s;
58          s = (hour % 12 == 0) ? String.format("   %02d:%02d:%02d ", 12,
                minute, second) :
59                  String.format("   %02d:%02d:%02d ", hour % 12, minute,
                        second);
60          s += (hour < 12) ? "AM" : "PM";
61          return s;
62      }
63
64      public void incrementHour() {
65          if (++hour >= 24) hour = 0;
66      }
67
68      public void incrementMinute() {
69          if (++minute >= 60) {
70              minute = 0;
71              incrementHour();
72          }
73      }
74
```

```
75      public void incrementSecond() {
76          if (++second >= 60) {
77              second = 0;
78              incrementMinute();
79          }
80      }
```

Listing 3: TestTime.java L33-L46

```
1       // My Test for new increment methods
2           System.out.println("-----NEW TESTS FOR INCREMENT-----");
3           MyTime t7 = new MyTime(1,2,3);
4           System.out.printf("t7 is initialized as %s\n",t7.toUniversalString
                ());
5           t7.incrementHour();
6           System.out.printf("  after t7.incrementHour(), t7 is %s\n", t7.
                toUniversalString());
7           t7.incrementMinute();
8           System.out.printf("  after t7.incrementMinute(), t7 is %s\n", t7.
                toUniversalString());
9           t7.incrementSecond();
10          System.out.printf("  after t7.incrementSecond(), t7 is %s\n", t7.
                toUniversalString());
11          MyTime t8 = new MyTime(23, 59, 59);
12          System.out.printf("t8 is initialized as %s\n",t8.toUniversalString
                ());
13          t8.incrementSecond();
14          System.out.printf("  after t8.incrementSecond(), t8 is %s\n", t8.
                toUniversalString());
```

1.5 总结与收获

本题目较为简单，但其中蕴含的方法间互相调用以提高代码复用率的思想让我收益良多。

本题中的输出涉及大量关于细节，包括：如何让代码的逻辑更舒服更“美”、java 字符串的 format 形式等等，这些细节虽然加起来不到三五行，却占据了我 50% 以上的时间，尤其是对代码逻辑的优化和打磨需要反复斟酌。

题目 2：创建自己的 GUI 图形

2.1 题目

在阅读 TestDraw、DrawPanel 和 MyLine 类定义的基础上，为 MyCircle 和 MyRectangle 添加相应的代码，以便让 TestDraw 程序可以绘制圆形和长方形。

1. 创建上面提到的两个类型，包括必要的该类型的数据成员、构造方法（可以利用重载技术，使该类型具有多种构造能力）、以及必不可少的一个成员方法：

```
1        public void draw(Graphics g)
```

   上面成员方法中的参数 g 为类型 java.awt.Graphics，使用这个类来绘制所需要的图形，该类型的使用可以在 JDK API 中查找。注意：

   - DrawPanel 的构造函数中提供了对 MyCircle 和 MyRectangle 类型数组的支持。
   - 尽可能为 MyCircle 类和 MyRectangle 类提供多样的构造函数，体会重载以及理解构造函数的意义（构造函数之间通过 this 传递的方式也要多练）。

2. 不难注意到上面的代码复用率极低，大刀破斧的修改 DrawPanel 类的数据成员、构造函数等，使其具有可以接收绘制任何（包括已有的和未来的）具有相同行为的图形类型。为了达到代码复用的目的，可能修改的不止 DrawPanel 类。

在实验报告中书写本题的设计环节部分时，请使用 UML 的类图进行阐述。

2.2 题目分析

本题分为两部分

2.2.1 创建类

- **Field**. 参考几何学知识，用三个 int 变量计算圆心坐标（x_position）和半径（y_position）以确定 MyCircle 的位置，用两个 Color 类分别表示线颜色和填充颜色。MyRectangle 的位置则需要四个 int 变量表示左下角和右上角坐标（或者说 x 和 y 坐标的上界和下界）。

- **Constructor**. MyCircle 和 MyRectangle 类的构造函数编写过程和 MyTime 的过程类似，比较枯燥乏味，区别仅限于一个是自己构思，一个是参考测试代码，故在此略过，细节可以查看 2.4 的主干代码展示和附录的所有代码展示；
- **Other Method**. 这部分需要调用 Graphics 的方法来编写类的 draw 方法，一些细节值得注意：
  - Panel 的坐标原点在左上方，这和一般的思路可能有差异；
  - 查询 Graphics 的方法可知，绘制椭圆（圆形）的方法 Oval 是通过右上角点的坐标、短轴和长轴对圆定位的，这和现实生活的习惯不符；
- **Test**. 对已有的 TestDraw 类修改可以使其测试 MyCircle 类和 MyRectangle 类

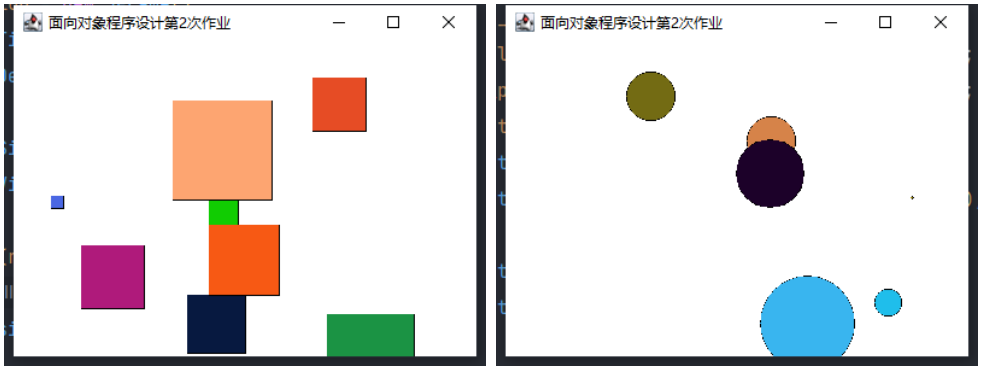
2.2.2 提高代码复用性

分析 DrawPanel 的代码，尤其是针对不同类的处理程序，并比较 MyLine, MyCircle 和 MyRectangle 三个类不难发现，有大量重复代码，所以可以设计一个 MyShape 作为父类，考虑到一般不会出现未实例化的 MyShape，且每个 MyShape 的子类都会有自己对 draw 的方法实现，而目前并未看到除此之外他们的关联，这里暂时把 MyShape 定义为抽象类。

这样就可以取消冗余而丑陋的利用 final int 和 case 作为选择，而转为直接对 MyShape 数组操作，以后新的图形类型只需 extends MyShape 即可。除此之外当然也需要使已定义的图形类型继承 MyShape.

考虑到题目三的便利，这里再将 MyShape 的 draw 函数剥离出来，定义为 Drawable 接口，从而实现对任一实现了 Drawable 的类的绘制功能。

2.3 结果展示



(a) MyRectangle 测试绘制结果　　(b) MyCircle 测试绘制结果

图 4: UML 图

2.4 主干代码说明

Listing 4: MyCircle.java L5-L86

```
1
2 public class MyCircle extends MyShape {
3         private int x_position;
4         private int y_position;
5         private int radix;
6         private Color context_Color;
```

```
7         private Color line_Color;
8
9         public MyCircle() {
10                x_position = 0;
11                y_position = 0;
12                radix = 0;
13                line_Color = Color.black;
14                context_Color = Color.white;
15        }
16
17        public MyCircle(int r) {
18                this();
19                radix = r;
20        }
21
22        public  MyCircle(int r, Color cc) {
23                this(r);
24                context_Color = cc;
25        }
26
27        public  MyCircle(int r, Color cc, Color lc) {
28                this(r, cc);
29                line_Color = lc;
30        }
31
32        public MyCircle(int x, int y, int r) {
33                this(r);
34                x_position = x;
35                y_position = y;
36        }
37
38        public MyCircle(int x, int y, int r, Color cc) {
39                this(x, y, r);
40                context_Color = cc;
41        }
42
```

```
43          public MyCircle(int x, int y, int r, Color cc, Color lc) {
44                  this(x, y, r, cc);
45                  line_Color = lc;
46          }
47
48          // Generate a circle taking line as diameter, just as an example,
               so no more Color-related method.
49          public MyCircle(MyLine line){
50                  this(line.getX1()+ line.getX2() >>1, line.getY1()+ line.
                       getY2()>>1,
51                                  line.getLength()>>1, line.getColor());
52          }
53
54          public int getX_position() {
55                  return x_position;
56          }
57
58          public int getY_position() {
59                  return y_position;
60          }
61
62          public int getRadix() {
63                  return radix;
64          }
65
66          public void setContext_Color(Color cc){
67                  context_Color = cc;
68          }
69
70          public Color getContext_Color() {
71                  return context_Color;
72          }
73
74          public Color getLine_Color() {
75                  return line_Color;
76          }
```

```
77
78          public void draw(Graphics g) {
79                  g.setColor(line_Color);
80                  g.drawOval(x_position - radix, y_position - radix, 2*radix,
                        2*radix);
81                  g.setColor(context_Color);
82                  g.fillOval(x_position - radix, y_position - radix, 2*radix,
                        2*radix);
83          }
```

Listing 5: MyRectangle.java L5-L71

```
1  public class MyRectangle extends MyShape implements calcAreable {
2          private int x_Low;
3          private int y_Low;
4          private int x_High;
5          private int y_High;
6          Color context_Color;
7          Color line_Color;
8
9          public MyRectangle() {
10                 x_Low = 0;
11                 y_Low = 0;
12                 x_High = 0;
13                 y_High = 0;
14                 line_Color = Color.black;
15                 context_Color = Color.white;
16         }
17
18         public MyRectangle(int x2, int y2) {
19                 this();
20                 x_High = x2;
21                 y_High = y2;
22         }
23
24         public MyRectangle(int x2, int y2, Color cc) {
25                 this(x2, y2);
26                 context_Color = cc;
```

```
27          }
28
29          public MyRectangle(int x2, int y2, Color cc, Color lc) {
30                  this(x2, y2, cc);
31                  line_Color = lc;
32          }
33
34          public MyRectangle(int x1, int y1, int x2, int y2) {
35                  this(x2, y2);
36                  x_Low = x1;
37                  y_Low = y1;
38          }
39
40          public MyRectangle(int x1, int y1, int x2, int y2, Color cc) {
41                  this(x1, y1, x2, y2);
42                  context_Color = cc;
43          }
44
45          public MyRectangle(int x1, int y1, int x2, int y2, Color cc, Color
                lc) {
46                  this(x1, y1, x2, y2, cc);
47                  line_Color = lc;
48          }
49
50          // Generate a rectangle taking line as diagonal just as an example,
                so no more Color-related method.
51          public MyRectangle(MyLine line) {
52                  this(Math.min(line.getX1(), line.getX2()), Math.min(line.
                        getY1(), line.getY2()), Math.max(line.getX1(), line.
                        getX2()), Math.max(line.getY1(), line.getY2()), line.
                        getColor());
53          }
54
55          public MyRectangle(MyCircle circle) {
56                  this(circle.getX_position() - circle.getRadix(), circle.
                        getY_position() - circle.getRadix(), circle.
```

```
                    getX_position() + circle.getRadix(), circle.
                    getY_position() + circle.getRadix(), circle.
                    getContext_Color(), circle.getLine_Color());
57          }
58
59          public void draw(Graphics g) {
60                  g.setColor(line_Color);
61                  g.drawRect(x_Low, y_Low, x_High - x_Low, y_High - y_Low);
62                  g.setColor(context_Color);
63                  g.fillRect(x_Low, y_Low, x_High - x_Low, y_High - y_Low);
64          }
```

Listing 6: MyShape.java

```
1  package hw2.p2;
2
3  abstract class MyShape implements Drawable {
4
5  }
```

Listing 7: Drawable.java

```
1  package hw2.p2;
2
3  import java.awt.*;
4
5  public interface Drawable {
6      void draw(Graphics g);
7  }
```

Listing 8: DrawPanel.java

```
1  package hw2.p2;
2
3  import javax.swing.JPanel;
4  import java.awt.*;
5
6
7  public class DrawPanel extends JPanel {
```

```
8
9       private static final long serialVersionUID = 1L;
10      private Object[] objects;
11
12      public DrawPanel() {
13          setBackground(Color.BLACK);
14      }
15      public DrawPanel(MyShape[] s) {
16          setBackground(Color.WHITE);
17          objects = s;
18      }
19
20      // @TODO deal with the following things.
21      public DrawPanel(Object o){
22          setBackground(Color.WHITE);
23          objects = new Object[1];
24          objects[0] = o;
25      }
26
27
28      public void paintComponent(Graphics g) {
29          super.paintComponent(g);
30          if (objects == null){
31              g.drawString("在DrawPanel的构造函数中，你传递的引用参数是NULL。
                    ", 50, 50);
32              return;
33          }
34
35          if (!objects.getClass().isArray()) {
36              g.drawString("在DrawPanel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);
37              return;
38          }
39          for (Object o : objects){
40              if(o instanceof Drawable)
41                  ((Drawable)o).draw(g);
```

```
42              else g.drawString(objects.getClass().getName() + "doesn't
                    implements Drawable",50,50);
43          }
44
45      }
46  }
```

2.6 总结和收获

本题更像作为作业 1 和作业 3 的衔接，第一个任务以更大的可扩展性让我练习了构造函数的设计方法，这让我对利用 this() 在构造函数间相互调用更加得心应手。而通过查阅官方文档了解 Graphics 的类方法并上手使用则锻炼了我阅读文档的能力。

第二个任务虽然可以只单纯地抽象一层，即设计一个父类囊括所有图形，但看过作业三后就知道这里如果再抽象一层接口出来会更好，即 DrawPanel 不用局限于 MyShape 类。但这样设计需要考虑一个新的问题：MyShape 类是否还有存在的必要？经过思考后我的解决方案是保留 MyShape 类，因为之后可能会添加其他的图形类的共性。这种抽象几层和类存在必要性的斟酌对我对面向对象设计的理解很有帮助。

题目 3. 接口的威力

3.1 题目

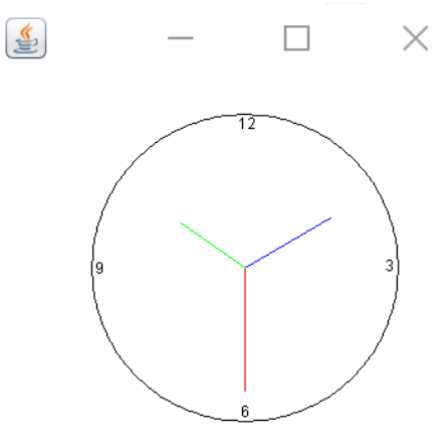
3.1.1 任务一

题目 2 中的图形类型的行为能力都是从绘制角度考虑的，但是一个图形类型除了具有被绘制的特点外，还有几何特征，比如面积。解决这个问题看似很简单：为每个图形类型增加一个求解面积的成员方法就可以了。但是会发现：并不是所有能够绘制的图形都具有求解面积的行为，比如 MyLine。那如何能够让应该具有面积特征的图形有求解面积的行为，不具有面积特征的图形就不具有求解面积的行为，同时还不破坏它们具有可以被绘制的多态效果的现状呢？Java 的接口类型可以助力完成上面的目标。本题第 1 个任务：请为该题目修改题目 2 中的图形类型，并绘制出新的图形类型之间的结构图。

扩展内容：再大胆一点…如何让 DrawPanel 类型可以绘制更多的东西（并不局限于图形类型的对象）？怎么修改 DrawPanel 中的数据成员类型以及构造方法？怎么定义类型使其能在 DrawPanel 上进行绘制？

3.1.2 任务二

依然尝试使用接口，比如创建了一个 MyTime 类型（这个类型第 1 个作业中被创建过），在 DrawPanel 上就可以绘制一个根据 MyTime 对象中的数据所表现出来的一个时钟。（DrawPanel 原有的绘制其他图形的能力应该依然保持。）如下图所示:



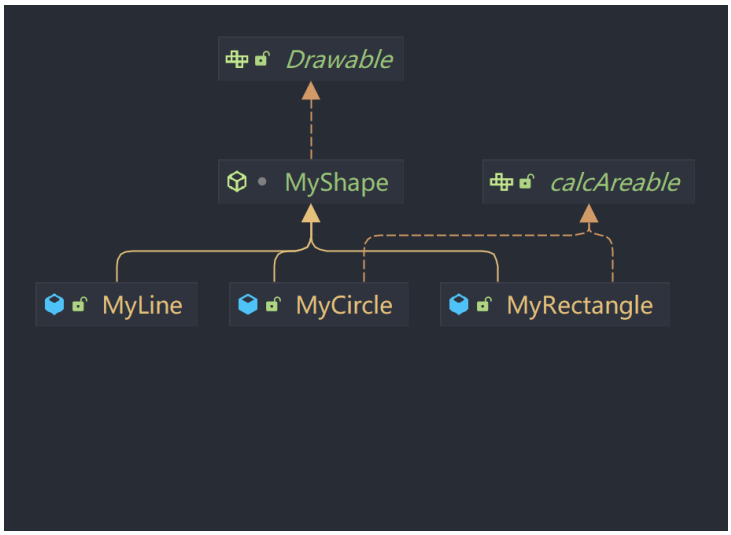
3.2 题目分析

本题任务一中的扩展内容我们已经在问题 2 中顺带解决，即创建一个 Drawable 接口，

3.2.1 面积接口与实现

对于可计算面积的图形也可以创建一个 calcAreable 接口以实现，接口内应定义返回面积的函数 double area();

在我们已定义的类中 MyRectangle 和 MyCircle 需要实现这一接口如下




```
public class MyRectangle extends MyShape implements calcAreable {
    // .....
    @Override
    public double area() {
                return (x_High - x_Low)*(y_High - y_Low);
        }
}
```

```
public class MyCircle extends MyShape implements calcAreable {
    // .....
    @Override
        public double area() {
                return Math.pow(radix,2)*Math.PI;
        }
}
```

3.2.2 绘制时钟

多亏了提前看到第三题从而在第二题进行了二次抽象，依照已有的类和接口不难得出，我们只需将已经写好的 MyTime 类实现 Drawable 接口。而根据题目，时钟的绘制分为三部分

1. 绘制圆形外壳. 这一步只需在以窗口的中心为圆心画圆即可
2. 标出四个时钟节点. 这一步需要用到 Graphics 的 drawString 方法，为使时钟更加美观，需要多次调整参数；
3. 绘制三条直线. 这里为了方便我们绘制新建一个 MyLine 的构造方法，参考极坐标系下画直线的过程，如下

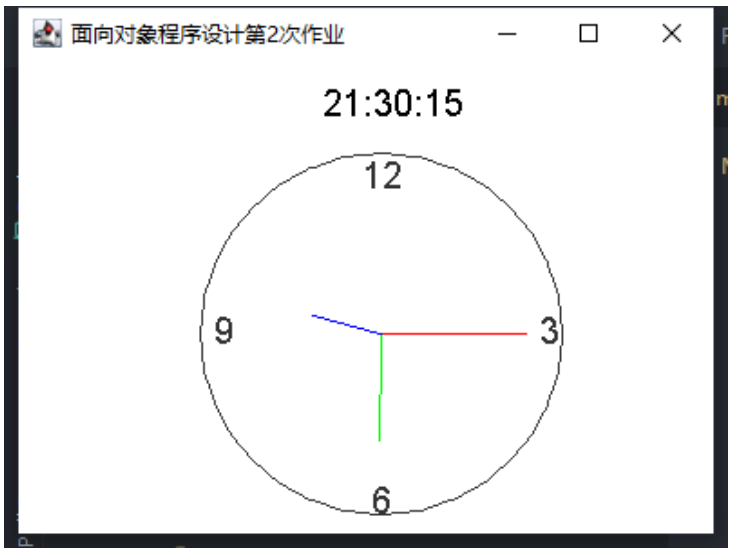
```
1    public MyLine(int x, int y, int length, double rad, Color c) {
2        this(x, y, x + (int) (length * Math.cos(rad)), y + (int) (
             length * Math.sin(rad)),c);
3    }
```

随后经过数学计算即可得到每个指针对应的参数和其数值大小的关系，为了使模拟更加逼真，我在更高级的时间角度上加了更低角度的偏差。

4. 为了便于展示，在时钟的上方再用 drawString 打印出时间.

3.3 运行结果展示

时钟运行结果如下



3.4 主干代码展示

```
1  public class MyTime implements Drawable {
2      // ....
3      @Override
4      public void draw(Graphics g) {
5          final int CENTER_X = 200, CENTER_Y = 150, RADIX = 100, FONT_SIZE =
               20;
6          g.drawOval(CENTER_X - RADIX, CENTER_Y - RADIX, 2 * RADIX, 2 * RADIX
               );
7
8          Font font = new Font("Arial", Font.PLAIN, FONT_SIZE);
9          g.setFont(font);
10         g.drawString("12", CENTER_X - FONT_SIZE / 2, CENTER_Y - RADIX +
               FONT_SIZE);
11         g.drawString("3", CENTER_X + RADIX - FONT_SIZE / 2 - 2, CENTER_Y +
               FONT_SIZE / 3);
12         g.drawString("6", CENTER_X - FONT_SIZE / 4, CENTER_Y + RADIX);
13         g.drawString("9", CENTER_X - RADIX + FONT_SIZE / 2 - 2, CENTER_Y +
               FONT_SIZE / 3);
14
15         double second_angle = (float) second / 30 * Math.PI;
16         MyLine second_line = new MyLine(CENTER_X, CENTER_Y, 80,
               second_angle - Math.PI / 2, Color.red);
17         second_line.draw(g);
18
19         double minute_angle = (float) minute/ 30 * Math.PI + second_angle /
                60;
20         MyLine minute_line = new MyLine(CENTER_X, CENTER_Y, 60,
               minute_angle - Math.PI / 2, Color.green);
21         minute_line.draw(g);
22
23         double hour_angle = (float)hour/6 *Math.PI + minute_angle / 12;
24         MyLine hour_line = new MyLine(CENTER_X, CENTER_Y, 40, hour_angle -
               Math.PI / 2, Color.blue);
25         hour_line.draw(g);
26
```

```
27          g.setColor(Color.black);
28          g.drawString(this.toUniversalString(),CENTER_X - RADIX/2, CENTER_Y
                - RADIX - FONT_SIZE );
29      }
30  }
```

3.5 总结和收获

在理解和掌握了接口的用法和思想后，不难发现本题的大体框架其实很简单，复杂的是本题中具体处理上的细节，包括 Font 类的创建，drawString 的参数，时钟怎么绘制更好看等等。

The Basic Idea of Java OOP

考虑到作业需要的是一份内容总结，而非讲义，故内容编排上和教学顺序有一定出入。

0x00 基本概念

重载（overload）：作为一门现代化的语言，Java 支持方法重载，即：只要方法名和参数类型不全相同即可视作一个新的方法。重载方法的原则遵循就近原则。

强制类型转换（cast）：Java 也支持变量强制类型转换，注意：这种转换只是一种对编译器的“欺骗”——即告诉编译器这一块为类型转换的目标类型。

Java 中一切非原始类型的都以引用类型在程序中存在。

Java 中每个引用变量有两个类型：

- 静态类型：声明时定义的类型，在编译时确定并检查，也就是说 cast 改变的是那一块的静态类型。
- 动态类型：变量实际指向的类型，在运行时确定。

Java 中用 final 关键字表示常量

0x01 类

Java 中的一切函数（方法）、变量和代码都是在一个个类中。

自然而然地，由基本的面向对象思想，一个类可以被另一个类所继承，Java 中用 extends 关键词。

```
1    public subClassA extends parentClassB{
2        // ...
3    }
```

子类会继承父类的所有变量和方法，可以通过 super 调用形成独一无二的“is a”关系

每个类都有变量（Field）和函数（Method）两部分，类中成员的声明需要访问权限修饰符，修饰符主要在其他类调用该类中的成员时确认合法性，Java 中共有四种修饰符，各自含义如下：

访问范围	private	friendly(默认)	protected	public
同一个类	√	√	√	√
同一个包中的其他类	×	√	√	√
不同包中的子类	×	×	√	√
不同包中的非子类	×	×	×	√

abstract& final

abstract:

- abstract 修饰的方法可以不定义，只能存在于 abstract 类中
- abstract 修饰的类不能被直接实例化，类中可以存在非 abstract 方法

final: final 修饰的类不能被继承，final 修饰的方法也不能进行修改。

0x02 接口

由于 Java 只支持类对类的单继承，为了使继承更加地灵活，除了类，Java 中还存在另一种和类相似的类型：接口。接口需要定义方法，但不实现方法 (不过随着 Java 版本的更新也有了 default 的方法实现)。

当一个类需要继承这个接口时，声明类时采用 implements 关键词，用法与 extends 类似。但一个类可以继承多个接口，接口与接口用逗号隔开。

```
1 public subClassA extends parentClassB implements InterfaceA, InterfaceB{
2     // ...
3 }
```

如果一个类继承了某个接口，则它一定要实现接口中的所有方法。

类和接口不存在 is a 的关系。

考虑到接口可以独立作为一个类型，也可以被实现，后文的叙述中涉及继承的部分默认也适用于继承自接口。

0x03 继承

继承的思想很好理解和接受，但继承中存在大量细节需要说明和理解记忆。

3.0 Constructor

子类的构造方法的第一句必须是父类的构造方法，这一句也可以通过调用子类的其他构造方法以实现。

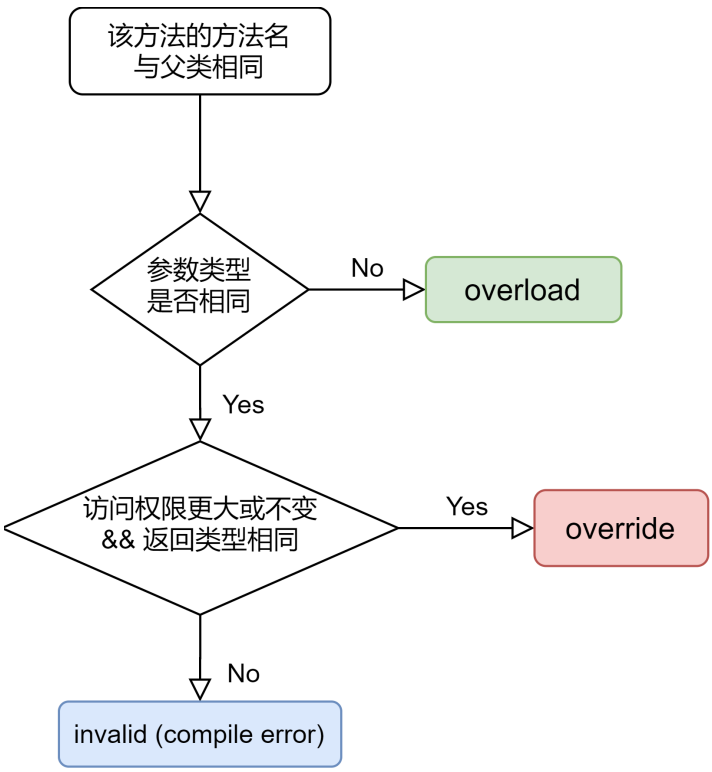
如果构造方法的第一句和上述不符，编译器会在第一句插一条父类的默认构造方法 super() ，如果父类没有默认的构造方法此处会编译报错。

3.1 Override

当子类中的某个**方法**和父类的方法名相同，修饰符访问范围更开放，且参数类型也相同时，我们称该方法覆盖（override）了父类中的原方法，此时若想通过子类调用父类方法只能通过 super()。

override 流程

1. 当且仅当**方法**参数类型和方法名均相同时会被认为是 override，开始检查 override 的合法性
2. 当方法名和参数类型均相同，但返回值不同或访问权限更小，这个函数不合法，会出现编译错误



add & override

变量中不存在覆盖的概念，如果变量出现重名，会新增一个变量。

如果出现重载，会新增一个方法。

3.2 Check& Selection

从编译的角度看：在编译过程中需要检查程序中每一句的合法性，这时只能采用静态类型确认，对静态类型来说不合法就会报错。

cast & type

有了继承的概念，我们可以先对 cast 的概念在继承的角度作进一步的说明。

编译时：当两个对象的类之间存在继承关系时可以 cast，否则会出现 compile error

运行时：如果对象的动态类型并非 cast 的结果，则会出现 exception. 从继承关系上看，向上转换可以毫无顾忌地执行（因为有“is a”关系），但向下转换时需要谨慎考虑。

Variable field

在 java 的类中，变量域依赖于静态类型而存在。

因此，如果需要不利用 super 访问父类的变量也可以利用 cast 转为父类再访问变量。

Method

除了实例方法外，类中的数据成员和类方法均依赖于对象的静态类型存在，也就是说在编译时就确定了要用哪个数据/方法。

实例方法比较特殊，采用一种动态选择的机制，在运行时才会依据引用变量指向的实例对象类型选择要执行的方法。

0x04 Examples

上述的概念尤其是 3.2 部分存在一定的抽象性，没有陷进坑里过很难一次性想明白，故提供几个样例程序如下

Listing 9: Bird.java

```
1 package hw2.p4;
2
3 public class Bird {
```

```
4       public void gulgate(Bird b) {
5           System.out.println("BiGulBi");
6       }
7   }
8
9   class Falcon extends Bird {
10      public void gulgate(Falcon f) {
11          System.out.println("FaGulFa");
12      }
13
14      public static void main(String[] args) {
15          Bird bird = new Falcon();
16          Falcon falcon = (Falcon) bird;
17          bird.gulgate(falcon);
18          falcon.gulgate(falcon);
19      }
20  }
21  // 运行结果为
22  // BiGulBi
23  // FaGulFa
24  // Overload is not override! 方法选择会遵循“就近原则”
```

Listing 10: Dog.java

```
1   package hw2.p4;
2
3   interface Animal {
4       default void greet(Animal a) {
5           System.out.println("hello animal");
6       }
7
8       default void sniff(Animal a) {
9           System.out.println("sniff animal");
10      }
11
12      default void praise(Animal a) {
13          System.out.println("u r cool animal");
14      }
```

```
15 }
16
17 public class Dog implements Animal {
18     @Override
19     public void sniff(Animal a) {
20         System.out.println("dog sniff animal");
21     }
22
23     void praise(Dog a) {
24         System.out.println("u r cool dog");
25     }
26
27     public static void main(String[] args) {
28         Animal a = new Dog();
29         Dog d = new Dog();
30         a.greet(d);
31         a.sniff(d);
32         d.praise(d);
33         a.praise(d);
34         ((Dog) a).praise(d);
35     }
36 }
37
38 // 运行结果为:
39 // hello animal          greet() 没有被override
40 // dog sniff animal      sniff() 被override了, 动态选择
41 // u r cool dog          praise() 被overload了, 就近原则
42 // u r cool animal       praise() 没有被override, 依旧选择父类
43 // u r cool dog          (cast 会转换静态类型)
```

根据以上两个代码可以对下面这段代码有更好的理解

Listing 11: Supplier.java

```
1 package hw2.p4;
2
3
4 // CAST only change static type (lie to compiler)
5 class Contact {
```

```
6       public String a;
7       public String b;
8       String doStuff() {
9           return "howdy\n";
10      }
11
12  }
13  class Supplier extends Contact {
14      String doStuff() {
15          return "send money \n";
16      }
17
18      void doStuff(int a ){ }
19      String dome() {
20          return "me\n";
21      }
22
23      public String a;
24      public String c;
25
26      public static void main(String[] args) {
27          Supplier supplier = new Supplier();
28          supplier.a = "supplier's Supplier :: a\n";
29          ((Contact)supplier).a = "supplier's Contact :: a by supplier cast\n
                ";
30          supplier.b = "supplier's Contact :: b\n";
31          supplier.c = "supplier's Contact :: b\n";
32          System.out.println(supplier.doStuff() + supplier.a + supplier.b +
                supplier.c);
33
34          Contact contact = new Contact();
35          contact.a = "contact's Contact :: a\n";
36          contact.b = "contact's Contact :: b\n";
37          System.out.println(contact.doStuff() + contact.a + contact.b);
38
39          contact = supplier;
```

```
40          ((Supplier)contact).a = "supplier's Supplier :: new a by contact
                cast\n";
41          contact.a = "supplier's Contact :: new a\n";
42          contact.b = "supplier's Contact :: new b\n";
43          ((Supplier)contact).c = "supplier's Supplier :: new c by contact
                cast\n";
44          System.out.println(((Contact)supplier).doStuff() + supplier.a +
                supplier.b + supplier.c);
45          System.out.println(contact.doStuff() + contact.a + contact.b + ((
                Supplier)contact).doStuff() + ((Contact)contact).doStuff());
46      }
47  }
48  // 运行结果:
49  // send money
50  // supplier's Supplier :: a
51  // supplier's Contact :: b
52  // supplier's Contact :: b
53  //
54  // howdy
55  // contact's Contact :: a
56  // contact's Contact :: b
57  //
58  // send money
59  // supplier's Supplier :: new a by contact cast
60  // supplier's Contact :: new b
61  // supplier's Supplier :: new c by contact cast
62  //
63  // send money
64  // supplier's Contact :: new a
65  // supplier's Contact :: new b
66  // send money
67  // send money
```

0x05 Polymorphism

什么是多态?

从程序到进程周期的角度看，Java 中存在两种多态：

- 静态多态（编译时多态性）：同名方法可以有多个，在编译时能够确定执行同名方法中的哪一个，则称为编译时多态性.
- 动态多态（运行时多态）：由于覆盖的影响，在编译时不能确定，只能在运行时才能确定执行多个同名方法中的哪一个，则称为运行时多态性.

从类的角度看，Java 中也存在类多态：

- 方法重载：同一个类的同名方法可以重载；
- 子类重定义：子类可以重定义父类的成员，从而隐藏变量或者覆盖方法

附录源码汇总

每个目录内按照文件首字母排序

I src/hw2/p1

Listing 12: MyTime.java

```
1  package hw2.p1;
2
3  import hw2.p2.Drawable;
4  import hw2.p2.MyLine;
5
6  import java.awt.*;
7
8  public class MyTime implements Drawable {
9      int hour;
10     int minute;
11     int second;
12
13     MyTime() {
14         hour = 0;
15         minute = 0;
16         second = 0;
17     }
18
19     MyTime(int h) {
20         this();
21         hour = h;
22     }
23
24     public MyTime(int h, int m) {
25         this(h);
26         minute = m;
27     }
```

```
28
29      public MyTime(int h, int m, int s) {
30          this(h, m);
31          second = s;
32      }
33
34      public MyTime(MyTime t4) {
35          this(t4.hour, t4.minute, t4.second);
36      }
37
38      boolean ValidJudge(int num) {
39          return num < 0 || num >= 60;
40      }
41
42      public String toUniversalString() {
43          boolean flag = true;
44          String s = "";
45
46          if (ValidJudge(hour)) {
47              flag = false;
48              s += "hour must be 0-23\n";
49          }
50          if (ValidJudge(minute)) {
51              flag = false;
52              s += "minute must be 0-59\n";
53          }
54          if (ValidJudge(second)) {
55              flag = false;
56              s += "second must be 0-59\n";
57          }
58          if (flag) return String.format("   %02d:%02d:%02d", hour, minute,
                second);
59          else return s;
60      }
61
62      @Override
```

```
63      public String toString() {
64          String s;
65          s = (hour % 12 == 0) ? String.format("   %02d:%02d:%02d ", 12,
                minute, second) :
66                   String.format("   %02d:%02d:%02d ", hour % 12, minute,
                         second);
67          s += (hour < 12) ? "AM" : "PM";
68          return s;
69      }
70
71      public void incrementHour() {
72          if (++hour >= 24) hour = 0;
73      }
74
75      public void incrementMinute() {
76          if (++minute >= 60) {
77              minute = 0;
78              incrementHour();
79          }
80      }
81
82      public void incrementSecond() {
83          if (++second >= 60) {
84              second = 0;
85              incrementMinute();
86          }
87      }
88
89      @Override
90      public void draw(Graphics g) {
91          final int CENTER_X = 200, CENTER_Y = 150, RADIX = 100, FONT_SIZE =
                20;
92          g.drawOval(CENTER_X - RADIX, CENTER_Y - RADIX, 2 * RADIX, 2 * RADIX
                );
93
94          Font font = new Font("Arial", Font.PLAIN, FONT_SIZE);
```

```
95          g.setFont(font);
96          g.drawString("12", CENTER_X - FONT_SIZE / 2, CENTER_Y - RADIX +
                FONT_SIZE);
97          g.drawString("3", CENTER_X + RADIX - FONT_SIZE / 2 - 2, CENTER_Y +
                FONT_SIZE / 3);
98          g.drawString("6", CENTER_X - FONT_SIZE / 4, CENTER_Y + RADIX);
99          g.drawString("9", CENTER_X - RADIX + FONT_SIZE / 2 - 2, CENTER_Y +
                FONT_SIZE / 3);
100
101         double second_angle = (float) second / 30 * Math.PI;
102         MyLine second_line = new MyLine(CENTER_X, CENTER_Y, 80,
                second_angle - Math.PI / 2, Color.red);
103         second_line.draw(g);
104
105         double minute_angle = (float) minute/ 30 * Math.PI + second_angle /
                 60;
106         MyLine minute_line = new MyLine(CENTER_X, CENTER_Y, 60,
                minute_angle - Math.PI / 2, Color.green);
107         minute_line.draw(g);
108
109         double hour_angle = (float)hour/6 *Math.PI + minute_angle / 12;
110         MyLine hour_line = new MyLine(CENTER_X, CENTER_Y, 40, hour_angle -
                Math.PI / 2, Color.blue);
111         hour_line.draw(g);
112
113         g.setColor(Color.black);
114         g.drawString(this.toUniversalString(),CENTER_X - RADIX/2, CENTER_Y
                - RADIX - FONT_SIZE );
115     }
116 }
```

Listing 13: TestTime.java

```
1  package hw2.p1;
2
3  public class TestTime {
4      public static void main(String[] args) {
5          // Official Test
```

```
6           MyTime t1 = new MyTime();
7           MyTime t2 = new MyTime(2);
8           MyTime t3 = new MyTime(21, 34);
9           MyTime t4 = new MyTime(12, 25, 42);
10          MyTime t5 = new MyTime(t4);
11
12          System.out.println("Constructed with:");
13          System.out.println("t1: all arguments defaulted");
14          System.out.printf(" %s\n", t1.toUniversalString());
15          System.out.printf(" %s\n", t1.toString());
16          System.out.println("t2: hour specified; minute and second defaulted
                ");
17          System.out.printf(" %s\n", t2.toUniversalString());
18          System.out.printf(" %s\n", t2.toString());
19          System.out.println("t3: hour and minute specified; second defaulted
                ");
20          System.out.printf(" %s\n", t3.toUniversalString());
21          System.out.printf(" %s\n", t3.toString());
22          System.out.println("t4: hour, minute and second specified");
23          System.out.printf(" %s\n", t4.toUniversalString());
24          System.out.printf(" %s\n", t4.toString());
25          System.out.println("t5: MyTime object t4 specified");
26          System.out.printf(" %s\n", t5.toUniversalString());
27          System.out.printf(" %s\n", t5.toString());
28          // when initialize t6 with invalid values,please output error
                information
29          MyTime t6 = new MyTime(15, 74, 99);
30          System.out.println("t6: invalid values");
31          System.out.printf("%s\n", t6.toUniversalString());
32
33          // My Test for new increment methods
34          System.out.println("-----NEW TESTS FOR INCREMENT-----");
35          MyTime t7 = new MyTime(1,2,3);
36          System.out.printf("t7 is initialized as %s\n",t7.toUniversalString
                ());
37          t7.incrementHour();
```

```
38          System.out.printf("  after t7.incrementHour(), t7 is %s\n", t7.
                toUniversalString());
39          t7.incrementMinute();
40          System.out.printf("  after t7.incrementMinute(), t7 is %s\n", t7.
                toUniversalString());
41          t7.incrementSecond();
42          System.out.printf("  after t7.incrementSecond(), t7 is %s\n", t7.
                toUniversalString());
43          MyTime t8 = new MyTime(23, 59, 59);
44          System.out.printf("t8 is initialized as %s\n",t8.toUniversalString
                ());
45          t8.incrementSecond();
46          System.out.printf("  after t8.incrementSecond(), t8 is %s\n", t8.
                toUniversalString());
47      }
48  }
```

II src/hw2/p2

Listing 14: calcAreable.java

```
1  package hw2.p2;
2
3  public interface calcAreable {
4      double area();
5  }
```

Listing 15: Drawable.java

```
1  package hw2.p2;
2
3  import java.awt.*;
4
5  public interface Drawable {
6      void draw(Graphics g);
7  }
```

Listing 16: DrawPanel.java

```
1  package hw2.p2;
2
3  import javax.swing.JPanel;
4  import java.awt.*;
5
6
7  public class DrawPanel extends JPanel {
8
9      private static final long serialVersionUID = 1L;
10     private Object[] objects;
11
12     public DrawPanel() {
13         setBackground(Color.BLACK);
14     }
15     public DrawPanel(MyShape[] s) {
16         setBackground(Color.WHITE);
17         objects = s;
18     }
19
20     // @TODO deal with the following things.
21     public DrawPanel(Object o){
22         setBackground(Color.WHITE);
23         objects = new Object[1];
24         objects[0] = o;
25     }
26
27
28     public void paintComponent(Graphics g) {
29         super.paintComponent(g);
30         if (objects == null){
31             g.drawString("在DrawPanel的构造函数中，你传递的引用参数是NULL。
                   ", 50, 50);
32             return;
33         }
34
```

```
35          if (!objects.getClass().isArray()) {
36              g.drawString("在DrawPanel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);
37              return;
38          }
39          for (Object o : objects){
40              if(o instanceof Drawable)
41                  ((Drawable)o).draw(g);
42              else g.drawString(objects.getClass().getName() + "doesn't
                    implements Drawable",50,50);
43          }
44
45      }
46  }
```

Listing 17: MyCircle.java

```
1   package hw2.p2;
2
3   import java.awt.*;
4
5   public class MyCircle extends MyShape implements calcAreable {
6           private int x_position;
7           private int y_position;
8           private int radix;
9           private Color context_Color;
10          private Color line_Color;
11
12          public MyCircle() {
13                  x_position = 0;
14                  y_position = 0;
15                  radix = 0;
16                  line_Color = Color.black;
17                  context_Color = Color.white;
18          }
19
20          public MyCircle(int r) {
21                  this();
```

```
22              radix = r;
23          }
24
25          public  MyCircle(int r, Color cc) {
26              this(r);
27              context_Color = cc;
28          }
29
30          public  MyCircle(int r, Color cc, Color lc) {
31              this(r, cc);
32              line_Color = lc;
33          }
34
35          public MyCircle(int x, int y, int r) {
36              this(r);
37              x_position = x;
38              y_position = y;
39          }
40
41          public MyCircle(int x, int y, int r, Color cc) {
42              this(x, y, r);
43              context_Color = cc;
44          }
45
46          public MyCircle(int x, int y, int r, Color cc, Color lc) {
47              this(x, y, r, cc);
48              line_Color = lc;
49          }
50
51          // Generate a circle taking line as diameter, just as an example,
                so no more Color-related method.
52          public MyCircle(MyLine line){
53              this(line.getX1()+ line.getX2() >>1, line.getY1()+ line.
                    getY2()>>1,
54                                  line.getLength()>>1, line.getColor());
55          }
```

```
56
57          public int getX_position() {
58                  return x_position;
59          }
60
61          public int getY_position() {
62                  return y_position;
63          }
64
65          public int getRadix() {
66                  return radix;
67          }
68
69          public void setContext_Color(Color cc){
70                  context_Color = cc;
71          }
72
73          public Color getContext_Color() {
74                  return context_Color;
75          }
76
77          public Color getLine_Color() {
78                  return line_Color;
79          }
80
81          public void draw(Graphics g) {
82                  g.setColor(line_Color);
83                  g.drawOval(x_position - radix, y_position - radix, 2*radix,
                        2*radix);
84                  g.setColor(context_Color);
85                  g.fillOval(x_position - radix, y_position - radix, 2*radix,
                        2*radix);
86          }
87
88          @Override
89          public double area() {
```

```
90                  return Math.pow(radix,2)*Math.PI;
91          }
92  }
```

Listing 18: MyLine.java

```
1   package hw2.p2;
2
3   import java.awt.Graphics;
4   import java.awt.Color;
5   public class MyLine extends MyShape {
6           private int x1;
7           private int y1;
8           private int x2;
9           private int y2;
10          private Color color;
11
12          public MyLine(int x1, int y1, int x2, int y2, Color color) {
13                  this.x1 = x1;
14                  this.y1 = y1;
15                  this.x2 = x2;
16                  this.y2 = y2;
17                  this.color = color;
18          }
19
20          public MyLine(int x, int y, int length, double rad, Color c) {
21                  this(x, y, x + (int) (length * Math.cos(rad)), y + (int) (
                        length * Math.sin(rad)),c);
22          }
23
24          public void draw(Graphics g) {
25                  g.setColor(color);
26                  g.drawLine(x1, y1, x2, y2);
27          }
28
29          public int getX1() {
30                  return x1;
31          }
```

```
32
33          public int getX2() {
34                  return x2;
35          }
36
37          public int getY1() {
38                  return y1;
39          }
40
41          public int getY2() {
42                  return y2;
43          }
44
45          public Color getColor() {
46                  return color;
47          }
48
49          public int getLength() {
50                  return (int) Math.round(Math.sqrt(Math.pow((x1 - x2), 2) +
                        Math.pow((y1 - y2), 2)));
51          }
52  }
```

Listing 19: MyRectangle.java

```
1   package hw2.p2;
2
3   import java.awt.*;
4
5   public class MyRectangle extends MyShape implements calcAreable {
6           private int x_Low;
7           private int y_Low;
8           private int x_High;
9           private int y_High;
10          Color context_Color;
11          Color line_Color;
12
13          public MyRectangle() {
```

```
14                  x_Low = 0;
15                  y_Low = 0;
16                  x_High = 0;
17                  y_High = 0;
18                  line_Color = Color.black;
19                  context_Color = Color.white;
20          }
21
22          public MyRectangle(int x2, int y2) {
23                  this();
24                  x_High = x2;
25                  y_High = y2;
26          }
27
28          public MyRectangle(int x2, int y2, Color cc) {
29                  this(x2, y2);
30                  context_Color = cc;
31          }
32
33          public MyRectangle(int x2, int y2, Color cc, Color lc) {
34                  this(x2, y2, cc);
35                  line_Color = lc;
36          }
37
38          public MyRectangle(int x1, int y1, int x2, int y2) {
39                  this(x2, y2);
40                  x_Low = x1;
41                  y_Low = y1;
42          }
43
44          public MyRectangle(int x1, int y1, int x2, int y2, Color cc) {
45                  this(x1, y1, x2, y2);
46                  context_Color = cc;
47          }
48
49          public MyRectangle(int x1, int y1, int x2, int y2, Color cc, Color
```

```
            lc) {
50              this(x1, y1, x2, y2, cc);
51              line_Color = lc;
52          }
53
54          // Generate a rectangle taking line as diagonal just as an example,
                so no more Color-related method.
55          public MyRectangle(MyLine line) {
56              this(Math.min(line.getX1(), line.getX2()), Math.min(line.
                    getY1(), line.getY2()),
57                              Math.max(line.getX1(), line.getX2()), Math.
                                    max(line.getY1(), line.getY2()), line.
                                    getColor());
58          }
59
60          public MyRectangle(MyCircle circle) {
61              this(circle.getX_position() - circle.getRadix(), circle.
                    getY_position() - circle.getRadix(),
62                              circle.getX_position() + circle.getRadix(),
                                     circle.getY_position() + circle.
                                    getRadix(),
63                              circle.getContext_Color(), circle.
                                    getLine_Color());
64          }
65
66          public void draw(Graphics g) {
67              g.setColor(line_Color);
68              g.drawRect(x_Low, y_Low, x_High - x_Low, y_High - y_Low);
69              g.setColor(context_Color);
70              g.fillRect(x_Low, y_Low, x_High - x_Low, y_High - y_Low);
71          }
72
73          @Override
74          public double area() {
75              return (x_High - x_Low)*(y_High - y_Low);
76          }
```

```
77 }
```

Listing 20: MyShape.java

```
1  package hw2.p2;
2
3  abstract class MyShape implements Drawable {
4
5  }
```

Listing 21: TestDraw.java

```
1  package hw2.p2;
2
3  import javax.swing.*;
4  import java.awt.*;
5  import java.util.Random;
6
7  public class TestDraw {
8      public static int APP_WIDTH = 400;
9      public static int APP_HEIGHT = 300;
10
11     public static void main(String[] args) throws InterruptedException {
12         DrawPanel line_panel = new DrawPanel(generateLines());
13         DrawPanel circle_panel = new DrawPanel(generateCircles());
14         DrawPanel rec_panel = new DrawPanel(generateRectangles());
15         JFrame application = new JFrame();
16         application.setTitle("面向对象程序设计第2次作业");
17         application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18
19         application.setSize(APP_WIDTH, APP_HEIGHT);
20         application.setVisible(true);
21
22         application.add(rec_panel);
23         Thread.sleep(5000);
24         rec_panel.setVisible(false);
25
26         application.add(circle_panel);
27         Thread.sleep(5000);
```

```
28          circle_panel.setVisible(false);
29
30          application.add(line_panel);
31          Thread.sleep(5000);
32          line_panel.setVisible(false);
33      }
34
35      public static MyLine[] generateLines() {
36          Random randomNumber = new Random();
37          MyLine[] lines;
38          lines = new MyLine[5 + randomNumber.nextInt(5)];
39          for (int count = 0; count < lines.length; count++) {
40              int x1 = randomNumber.nextInt(APP_WIDTH);
41              int y1 = randomNumber.nextInt(APP_HEIGHT);
42              int x2 = randomNumber.nextInt(x1-60,x1+60);
43              int y2 = randomNumber.nextInt(y1-60,y1+60);
44              Color color = new Color(randomNumber.nextInt(256), randomNumber
                    .nextInt(256),
45                      randomNumber.nextInt(256));
46              lines[count] = new MyLine(x1, y1, x2, y2, color);
47          }
48          return lines;
49      }
50
51      public static MyCircle[] generateCircles() {
52          Random randomNumber = new Random();
53          MyLine[] lines = generateLines();
54          MyCircle[] circles = new MyCircle[lines.length];
55          for (int count = 0; count < lines.length; count++) {
56              Color color = new Color(randomNumber.nextInt(256), randomNumber
                    .nextInt(256),
57                      randomNumber.nextInt(256));
58              circles[count] = new MyCircle(lines[count]);
59              circles[count].setContext_Color(color);
60          }
61          return circles;
```

```
62      }
63
64      public static MyRectangle[] generateRectangles() {
65          MyCircle[] circles = generateCircles();
66          MyRectangle[] rectangles;
67          rectangles = new MyRectangle[circles.length];
68          for (int count = 0; count < rectangles.length; count++) {
69              rectangles[count] = new MyRectangle(circles[count]);
70          }
71          return rectangles;
72      }
73  }
```

Listing 22: TestDrawTimer.java

```
1   package hw2.p2;
2
3   import hw2.p1.MyTime;
4
5   import javax.swing.*;
6
7   public class TestDrawTimer {
8       public static int APP_WIDTH = 400;
9       public static int APP_HEIGHT = 300;
10
11      public static void main(String[] args) throws InterruptedException {
12          MyTime Test_Time =  new MyTime(21, 30, 15);
13          DrawPanel timer_panel = new DrawPanel(Test_Time);
14          JFrame application = new JFrame();
15          application.setTitle("面向对象程序设计第2次作业");
16          application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17          application.add(timer_panel);
18          application.setSize(APP_WIDTH, APP_HEIGHT);
19          application.setVisible(true);
20          Thread t = new Thread() {
21              public void run() {
22                  while (true) {
23                      try {
```

```
24                          Thread.sleep(1000); //设置绘制的时间间隔为 1 秒
25                      } catch (InterruptedException e) {
26                          System.err.println(e);
27                      }
28                      //正好也可以用来检验第 1 道题目中时钟类型的这个方法是否
                            正确
29                      Test_Time.incrementSecond();
30                      //更新绘制图形面板上的内容（也就是绘制的图像）
31                      timer_panel.updateUI();
32                  }
33              }
34          };
35          t.start();
36      }
37
38  }
```

参考文献

A. 内容参考

a. Java® Platform, Standard Edition & Java Development Kit Version 17 API Specification. [DB/OL], https://docs.oracle.com/en/java/javase/17/docs/api/index.html, 2022.

b. 原盛. “面向对象程序设计方法”课件, [Z/OL], 2022.

c. Josh Hug. PPT of Lec8&9 in CS61B(2021spring) from UC • Berkeley. [Z/OL], https://docs.google.com/presentation/d/1EUOpd9NXq28eEUqQMZoo_rRpxw8EekQ2LEp61U9b
2021.

B. LaTeX 代码参考

b. 【LaTeX】自用简洁模板（六）：学校作业

c. LaTeX 里「添加程序代码」的完美解决方案