Java 课程上机报告

上机题目：Java 的面向对象设计

小组名单：

张俊华 16030199025

李金鑫 16030199026

李天浩 16030199027

上机时间：2017/03/29 18:00-22:00
地点：EIII-204

实验目标：

掌握面向对象的编程方法；掌握 Java 的面向对象特性；掌握采用面向对象技术构建系统的一般方法。

实验要求：

1．练习 PPT 中的全部小练习，尝试对小练习中各部分进行修改，并观察修改后的执行效果。（自觉完成不需要交）

2．编写一个程序，要求如下：

1) 程序实现图形创建（模拟创建，如：用户输入 1，表示创建一个矩形；输入 2，表示创建一个圆；输入 3，表示创建一个三角形），并在创建时对该图形命名，最后在内存中保存这些创建的图形（最多 20 个）。
2) 程序提供针对名称的检索功能，即，根据用户提供的名称在保存的图形中查找图形并输出该图形的类型及创建序号。
3) 程序提供对图形的绘制功能（模拟绘制，如：绘制序号为 1，名称为教学楼的矩形图形，可以输出“1 矩形 教学楼”），即，根据用户输入的创建序号，顺序输出该序号之前，包括该序号的全部图形。
4) 构建所有图形的父类：Shape，该类中定义图形的基本属性及方法。
5) 构建基本图形类：矩形(Rectangle)、圆(Circle)、三角形(Triangle)。
6) 可通过多态实现对任意图形的绘制。
7) 定义静态方法，该方法可以对传入的对象实例进行判断，并输出该对象实例的类型。
8) 构建测试类，该类实现与用户的交互，向用户提示操作信息，并接收用户的操作请求。

程序应具有良好的类层次结构，良好的人机交互性能，即：程序应向用户提示功能说明，并可根据用户的功能选择，执行对应的功能，并给出带详细描述信息的最终执行结果。

一．题目分析：

本实验要求编写程序实现模拟图形的创建，按照题目要求，每个图形应该是一个对象，由对应的类实例化形成，这些类有着共同的一些属性，该该继承自同一个父类，同时需要继承父类的构造方法，但这些类图形的打印方法不同，应该复写父类的 print 方法。

我们还应该在父类中定义一些方法，使之能够获取和修改图形的序号和名称，避免对属性字段的直接访问。

输出查询结果时，我们通过下标 index 在数组中找到对应的对象，执行这个对象的 get 方法，实现程序的输出。

二．小组讨论内容：

张俊华：

1．实验环境：

IntelliJ IDEA 2017.1 Build #IU-171.3780.107

JRE: 1.8.0_112-release-736-b13 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 10 10.0

2. 实现过程：

先定义一个父类 graph，这个类是抽象的，它有 shape，index，name 这些公共字段；还有一个抽象方法 print

```
package Shapes;

/**
 * Created by 张俊华 on 2017/3/28.
 *
 * @author 张俊华.
 * @Time 2017/3/28 22:46.
 */
public abstract class Graph {

    public int graphShape;
    //图形形状
    public String graphName;
    //图形名称
    public int graphIndex;
    //图形序号
    public Graph(int graphIndex, int graphShape,
String graphName){
        this.graphIndex = graphIndex;
        this.graphShape = graphShape;
        this.graphName = graphName;
    }

    public abstract void print();
    public void setGraphShape(String name){
        this.graphName = name;
    }
    public void setGraphIndex(int n){
        this.graphIndex = n;
    }

}
```

创建子类。继承父类 graph，在每个子类中对 print 方法行行实现，

```
package Shapes;

/**
 * Created by 张俊华 on 2017/3/28.
 *
 * @author 张俊华.
 * @Time 2017/3/28 23:11.
 */
public class Circle extends Graph{
    public Circle(int graphIndex, int graphShape,
String graphName) {
        super(graphIndex, graphShape, graphName);
    }

    public void print(){
        System.out.println("圆形");
    }
}
```

用户交互：创建每一个图形时，获得用户输入的图形类型，图形名称，按照用户输入的图形类型，实例化对应的图形类，并将实例化的对象存入 graph 数组中，方便以后调用。

```
switch (graphShape) {
    case 1:
        graphs[index] = new Circle(index, graphShape,
graphName);
        break;
    case 2:
        graphs[index] = new Triangle(index,
graphShape, graphName);
        break;
    case 3:
        graphs[index] = new Rectangle(index,
graphShape, graphName);
}
```

新建一个 class 作为对列表功能的拓展

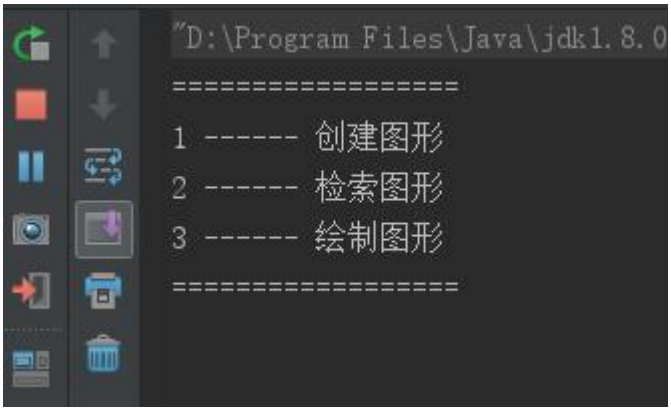
```
package action;

import Shapes.Circle;
import Shapes.Graph;
import Shapes.Rectangle;
import Shapes.Triangle;
import ui.UI;

/**
 * Created by 张俊华 on 2017/3/28.
 *
 * @author 张俊华.
 * @Time 2017/3/28 23:17.
 */
public class GraphList {
    private Graph graphs[] = new Graph[20];
    private int index = 0;
    private UI ui = new UI();

    public void appendGraphlist(int graphShape,
String graphName) {

        switch (graphShape) {
            case 1:
                graphs[index] = new Circle(index,
graphShape, graphName);
                break;
            case 2:
                graphs[index] = new Triangle(index,
graphShape, graphName);
                break;
            case 3:
                graphs[index] = new Rectangle(index,
graphShape, graphName);
        }
        this.index++;
    }
```

```
    public String getGraphShape(int index) {
        switch (graphs[index].graphShape) {
            case 1:
                return "圆形";
            case 2:
                return "三角形";

            case 3:
                return "矩形";
        }
        return "";
    }

    public String getGraphName(int index) {
        return graphs[index].graphName;
    }
}
```

与用户的交互全部在 main 方法中实现。

三. 程序结果：

初始菜单：



程序执行实例

```
==================
1 ------ 创建图形
2 ------ 检索图形
3 ------ 绘制图形
==================
1
1 ----- 圆形
2 ----- 三角形
3 ----- 矩形
请输入图形类型：
1
请输入图形名称：c1
创建成功!


==================
1 ------ 创建图形
2 ------ 检索图形
3 ------ 绘制图形
==================
3
请输入图形序号：0
圆形
c1
```

实验心得：

本次实验中我主要练习了类的定义和使用，学会了运用抽象类和抽象方法，练习了子类对父类的继承，以及多态的实现，感受到了面向对象的特性。不足之处在于：在图形类中定义图形种类这个属性是多余的，完全可以通过 JAVA 的 classtype 方法获得到类型名称。我意识到，JAVA 学习的过程还很漫长，我对 Java 的学习才刚刚开始，我会不断提高我的编程水平，在这条路上一直走下去。

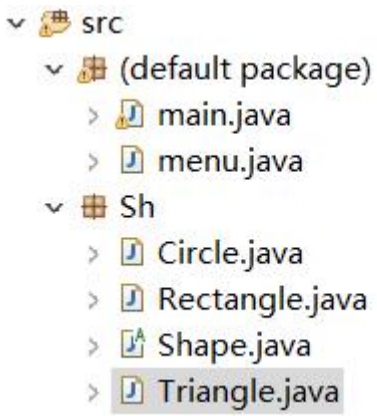
李金鑫：

1．实验环境：

IntelliJ IDEA 2017.1 Build #IU-171.3780.107
JRE: 1.8.0_112-release-736-b13 amd64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Windows 10 10.0

2．实现过程：

程序结构如下



首先定义 main 类，定义了存储图形的 shape 对象数组，定义了 menu 类实现打印交互界面，用户输入 n，从而进入不同的功能

```java
import java.util.Scanner;
public class main {
    public static void main(String[]args){
        Scanner in=new Scanner(System.in);
        int k=0;
        Shape[] Shapes=new Shape[20];
        int n;
        menu P=new menu();
        P.print();
    do{
        n=in.nextInt();
```

```java
public class menu {
    public void print()
    {
        System.out.println("1-----创建一个矩形");
        System.out.println("2-----创建一个圆形");
        System.out.println("3-----创建一个三角形");
        System.out.println("4-----绘制图形");
        System.out.println("5-----检索图形");
        System.out.println("6-----退出系统");
    }
}
```

在 Sh 包内定义了父类 Shape，包含名称 name，类型 type 两个参数，另外有一个抽象方法，在 Rectangle、Circle 和 Triangle 中进行了方法重写。

```java
package Sh;

public abstract class Shape {
    public String name;
    public int type;
    public abstract void print();
}

```

```java
package Sh;

public class Rectangle extends Shape{
    public void print(){
        System.out.print("矩形"+" ");
    }
}
```

```java
package Sh;

public class Circle extends Shape{
    public void print(){
        System.out.print("圆形"+" ");
    }
}
```

```java
package Sh;

public class Triangle extends Shape{
    public void print(){
        System.out.print("三角形"+" ");
    }
}
```

K 用来存储数组的数量，用户输入了 123 操作都是累加 k,输入名称，并且根据创建的不同自动生成类型 type。

```java
case 1:
    k++;
    Shapes[k]= new Rectangle();
    System.out.println("请输入图形名称");
    Shapes[k].name=in.next();
    Shapes[k].type=1;
    break;
case 2:
    k++;
    Shapes[k]= new Circle();
    System.out.println("请输入图形名称");
    Shapes[k].name=in.next();
    Shapes[k].type=2;
    break;
case 3:
    k++;
    Shapes[k]= new Triangle();
    System.out.println("请输入图形名称");
    Shapes[k].name=in.next();
    Shapes[k].type=3;
    break;
```

若输入了 4，则从头到尾循环调用 print 方法。

```
            break;
        case 4:
            System.out.println("请输入绘制序号");
            int s=in.nextInt();
            for (int i=1;i<=s;i++){
                System.out.print(i+" ");
                Shapes[i].print();
                System.out.println(Shapes[i].name);
            }
            break;
```

检索图形同样是循环利用.equals()方法比较。

```
        case 5:
            System.out.println("请输入图形的名称");
            String str=in.next();
            for (int i=1;i<=k;i++)
                if (Shapes[i].name.equals(str))
                {
                    Shapes[i].print();
                    System.out.println(i);
                }
            break;
    }
```

三．程序结果：

用户界面：

```
main (2) [Java Application] C:\Progra
1-----创建一个矩形
2-----创建一个圆形
3-----创建一个三角形
4-----绘制图形
5-----检索图形
6-----退出系统
```

创建功能：

```
main (2) [Java Application] C:\Progr
1-----创建一个矩形
2-----创建一个圆形
3-----创建一个三角形
4-----绘制图形
5-----检索图形
6-----退出系统
1
请输入图形名称
a
2
请输入图形名称
b
3
请输入图形名称
c
```

绘制功能：

```
请输入绘制序号
3
1 矩形 a
2 圆形 b
3 三角形 c
```

检索功能：

```
5
请输入图形的名称
b
圆形 2
```

李天浩：

1．实验环境：

IntelliJ IDEA 2017.1 Build #IU-171.3780.107
JRE: 1.8.0_112-release-736-b13 amd64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Windows 10 10.0

2. 实现过程：

```
public abstract class Shape {
    protected int id;
    public String Name;
    public void Init(int NewID,String NewName){
        id=NewID;
        Name=NewName;
    }
}
public class Rectangle extends Shape{
    public void Print(){
        System.out.println(id+" Rectangle"+" "+Name);
    }

}
public class Circle extends Shape{
    public void Print(){
        System.out.println(id+" Circle"+" "+Name);
    }
}
public class Triangle extends Shape{
   public void Print(){
        System.out.println(id+" Triangle"+" "+Name);
    }
}
```

这些是 Shape 类型和其子类

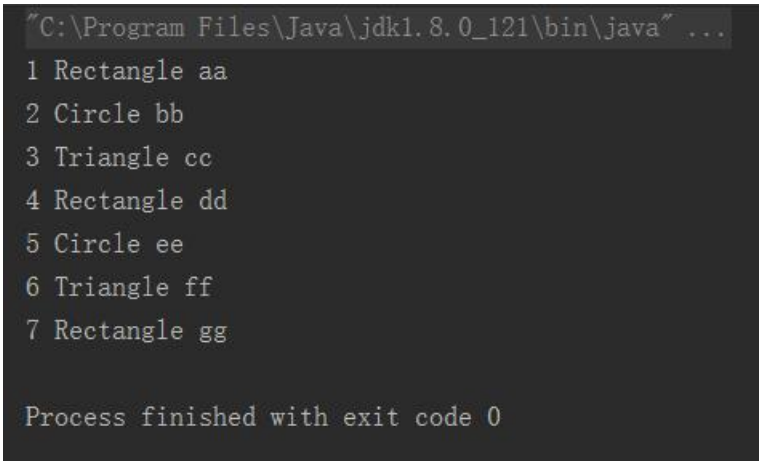
```
import java.util.*;
import base.*;


public class Container {
```

```
class pos{
    public int op,po;
    pos(){}
    pos(int x,int y){op=x;po=y;}
}

Vector<Rectangle>Rs;
Vector<Circle>Cs;
Vector<Triangle>Ts;
Vector<pos>index;
int cnt=1;
Container(){
    Rs=new Vector<>();
    Cs=new Vector<>();
    Ts=new Vector<>();
    index=new Vector<>();
    cnt=1;
}
public void add(int op,String Name){
    if(op==1){
        Rectangle newob=new Rectangle();
        newob.Init(cnt,Name);
        Rs. add(newob);
        index.add(new pos(op,Rs.size()));
    }else if(op==2){
        Circle newob=new Circle();
        newob.Init(cnt,Name);
        Cs.add(newob);
        index.add(new pos(op,Cs.size()));
    }else if(op==3){
        Triangle newob=new Triangle();
        newob.Init(cnt,Name);
        Ts.add(newob);
```

```
            index.add(new pos(op,Ts.size()));
        }
        cnt++;
    }
    public void FindbyName(String q) {
        for (pos c:index) {
            if (c.op==1) {
                Rectangle newob=(Rectangle) Rs.get(c.po-1);
                newob.Print();
                if(newob.Name.equals(q)) break;
            }
            else if(c.op==2){
                Circle newob=(Circle)Cs.get(c.po-1);
                newob.Print();
                if(newob.Name.equals(q))break;
            }else if(c.op==3){
                Triangle newob=(Triangle)Ts.get(c.po-1);
                newob.Print();
                if(newob.Name.equals(q))break;

            }

        }
    }
}
```

这里我使用了 java 的容器来对不同的类型分别存储，通过 index 来进行按输入顺序遍历

运行结果



四．实验心得：

张俊华：

本次实验中我主要练习了类的定义和使用，学会了运用抽象类和抽象方法，练习了子类对父类的继承，以及多态的实现，感受到了面向对象的特性。不足之处在于：在图形类中定义图形种类这个属性是多余的，完全可以通过 JAVA 的 classtype 方法获得到类型名称。我意识到，JAVA 学习的过程还很漫长，我对 Java 的学习才刚刚开始，我会不断提高我的编程水平，在这条路上一直走下去。

李金鑫：

本次实验中我主要掌握了父类和子类的联系方法，设置了两个包让程序结构有条理性，设置变量名称的时候更注重规范。在绘制时，用子类重写了父类方法。在检索时，明白了字符串的比较不能单纯的用“==”，今后的练习要提高编程速度，培养编程思想。

李天浩：

本次实验中我主要练习了类的定义和使用，学会了运用抽象类和抽象方法，练习了子类对父类的继承，以及多态的实现，感受到了面向对象的特性。也学习了 java 中容器的使用。对于输出类的名字，我通过网络尝试了很多方法，但是都没有成功，算是本次实验的一个遗憾吧。同时我也看到了 C++和 java 在 oop 方面的一些相似之处，这将帮我更好的学习 java 语言。