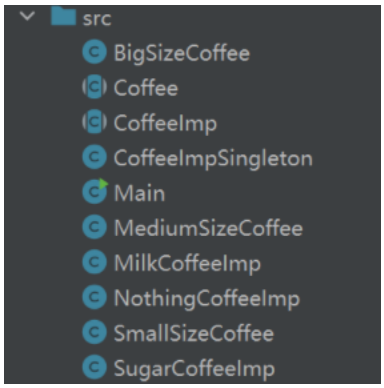
软件体系结构 作业14

22920212204392 黄勖

1　试用Bridge模式完成下列事情：饮料的杯子有大、中、小；行为有：加奶，加糖，啥都不加。

项目结构如下：



- **CoffeeImpSingleton**: 一个单例模式的类，用于管理不同的咖啡添加行为实例。
- **CoffeeImp**: 抽象类，定义了一个`pourCoffeeImp()`方法，用于具体描述添加行为。
- **Concrete Implementations (**`MilkCoffeeImp`，`SugarCoffeeImp`，`NothingCoffeeImp`**)**: 分别实现了`pourCoffeeImp()`方法，以打印不同的添加行为（加奶、加糖、什么都不加）。
- **Coffee**: 抽象基类，持有一个`CoffeeImp`实例，定义了`makeCoffee()`方法，用于开始制作咖啡的过程。
- **Size Specific Coffee Classes (**`BigSizeCoffee`，`MediumSizeCoffee`，`SmallSizeCoffee`**)**: 这些类继承自`Coffee`，并通过调用不同的添加行为实现具体的咖啡制作过程。
- **Main**: 包含main方法，初始化不同的咖啡添加行为，并创建不同大小和添加方式的咖啡实例进行制作。

具体代码：

CoffeeImpSingleton：

```
public class CoffeeImpSingleton {
    private static CoffeeImp coffeeImp;
    public CoffeeImpSingleton(CoffeeImp coffeeImpIn){
        coffeeImp = coffeeImpIn;
    }
    public CoffeeImp getCoffeeImp(){
        return coffeeImp;
    }
}
```

Coffee:

```
public abstract class Coffee {
    protected CoffeeImp coffeeImp;
    public void setCoffeeImp(CoffeeImp coffeeImpIn){
        coffeeImp = coffeeImpIn;
    }
    public CoffeeImp getCoffeeImp(){
        return coffeeImp;
    }
    public abstract void makeCoffee();
}
```

CoffeeImp:

```
public abstract class CoffeeImp {
    public abstract void pourCoffeeImp();
}
```

MilkCoffeeImp:

```
public class MilkCoffeeImp extends CoffeeImp{
    public void pourCoffeeImp(){
        System.out.println("加了美味的牛奶");
    }
}
```

SugarCoffeeImp:

```
public class SugarCoffeeImp extends CoffeeImp{
    public void pourCoffeeImp(){
        System.out.println("加了白砂糖");
    }
}
```

NothingCoffeeImp:

```
public class NothingCoffeeImp extends CoffeeImp{
    public void pourCoffeeImp(){
        System.out.println("什么都不加");
    }
}
```

BigSizeCoffee:

```
public class BigSizeCoffee extends Coffee{

    public void makeCoffee(){
        System.out.println("开始咖啡制作");
        for(int i = 1; i <= 3; ++i){
            System.out.println("加入咖啡");
            coffeeImp.pourCoffeeImp();
        }
        System.out.println("咖啡制作完成\n");
    }
}
```

MediumSizeCoffee:

```
public class MediumSizeCoffee extends Coffee{
   public void makeCoffee(){
     System.*out*.println("开始咖啡制作");
     for(int i = 1; i <= 2; ++i){
       System.*out*.println("加入咖啡");
       coffeeImp.pourCoffeeImp();
     }
     System.*out*.println("咖啡制作完成\n");
   }
 }
```

SmallSizeCoffee:

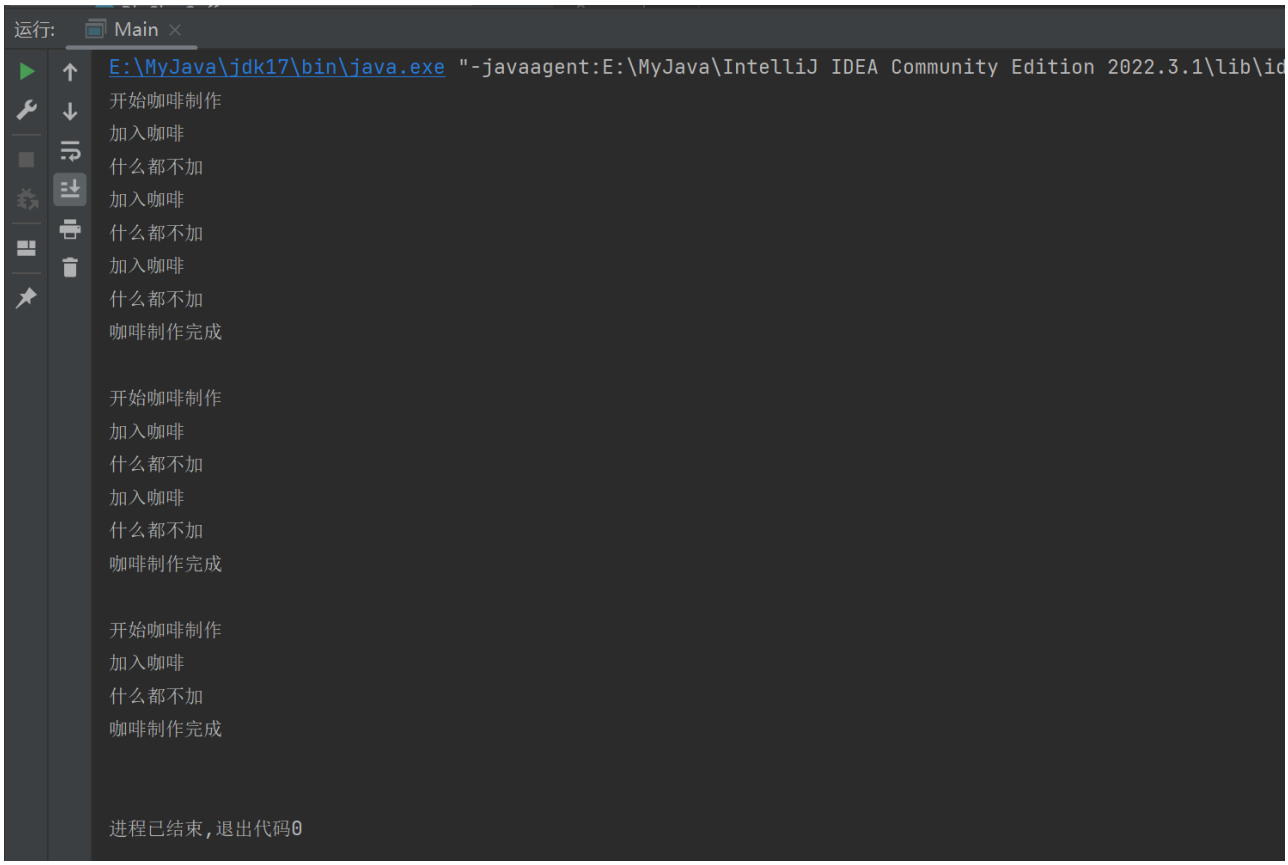
```
public class SmallSizeCoffee extends Coffee{
    public void makeCoffee(){
        System.out.println("开始咖啡制作");
        for(int i = 1; i <= 1; ++i){
            System.out.println("加入咖啡");
            coffeeImp.pourCoffeeImp();
        }
        System.out.println("咖啡制作完成\n");
    }
}
```

Main:

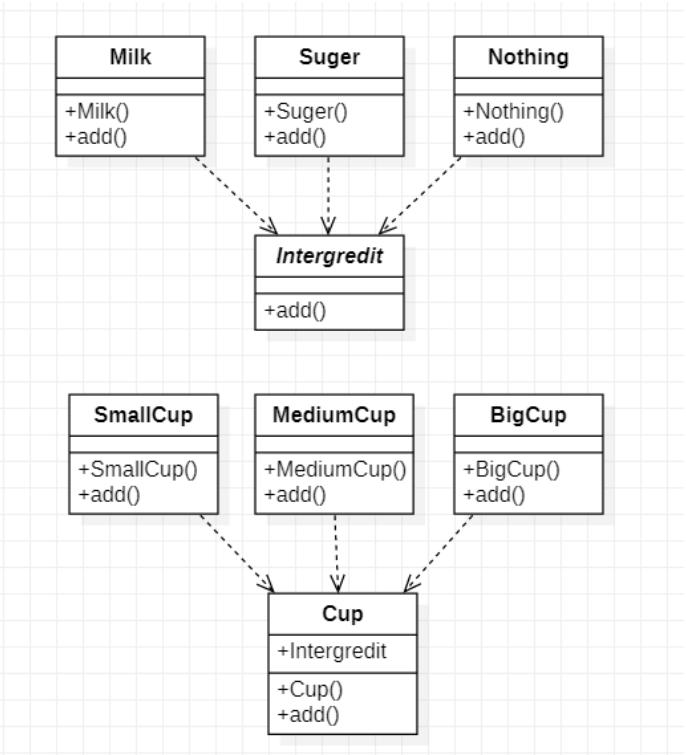
```
public class Main {
    public static void main(String[] args) {
        CoffeeImpSingleton milkCoffeeImpSingleton = new CoffeeImpSingleton(new
MilkCoffeeImp());
        CoffeeImpSingleton sugarCoffeeImpSingleton = new CoffeeImpSingleton(new
SugarCoffeeImp());
```

```
        CoffeeImpSingleton nothingCoffeeImpSingleton = new
CoffeeImpSingleton(new NothingCoffeeImp());

        //大杯加奶
        Coffee coffee1 = new BigSizeCoffee();
        coffee1.setCoffeeImp(milkCoffeeImpSingleton.getCoffeeImp());
        coffee1.makeCoffee();

        //中杯加糖
        Coffee coffee2 = new MediumSizeCoffee();
        coffee2.setCoffeeImp(sugarCoffeeImpSingleton.getCoffeeImp());
        coffee2.makeCoffee();

        //小杯不加
        Coffee coffee3 = new SmallSizeCoffee();
        coffee3.setCoffeeImp(nothingCoffeeImpSingleton.getCoffeeImp());
        coffee3.makeCoffee();

    }
}
```

运行结果如下：



可见答案如预期，正确


Milk
+Milk()
+add()
Suger
+Suger()
+add()
Nothing
+Nothing()
+add()
Intergredit
+add()
SmallCup
+SmallCup()
+add()
MediumCup
+MediumCup()
+add()
BigCup
+BigCup()
+add()
Cup
+Intergredit
+Cup()
+add()