模板方法模式

模式定义

模板方法模式在一个方法中定义了一个算法的骨架, 而将一些步骤延迟到子类中. 模板方法使得子类可以在不改变算法结构的前提下, 重新定义算法中的某些步骤

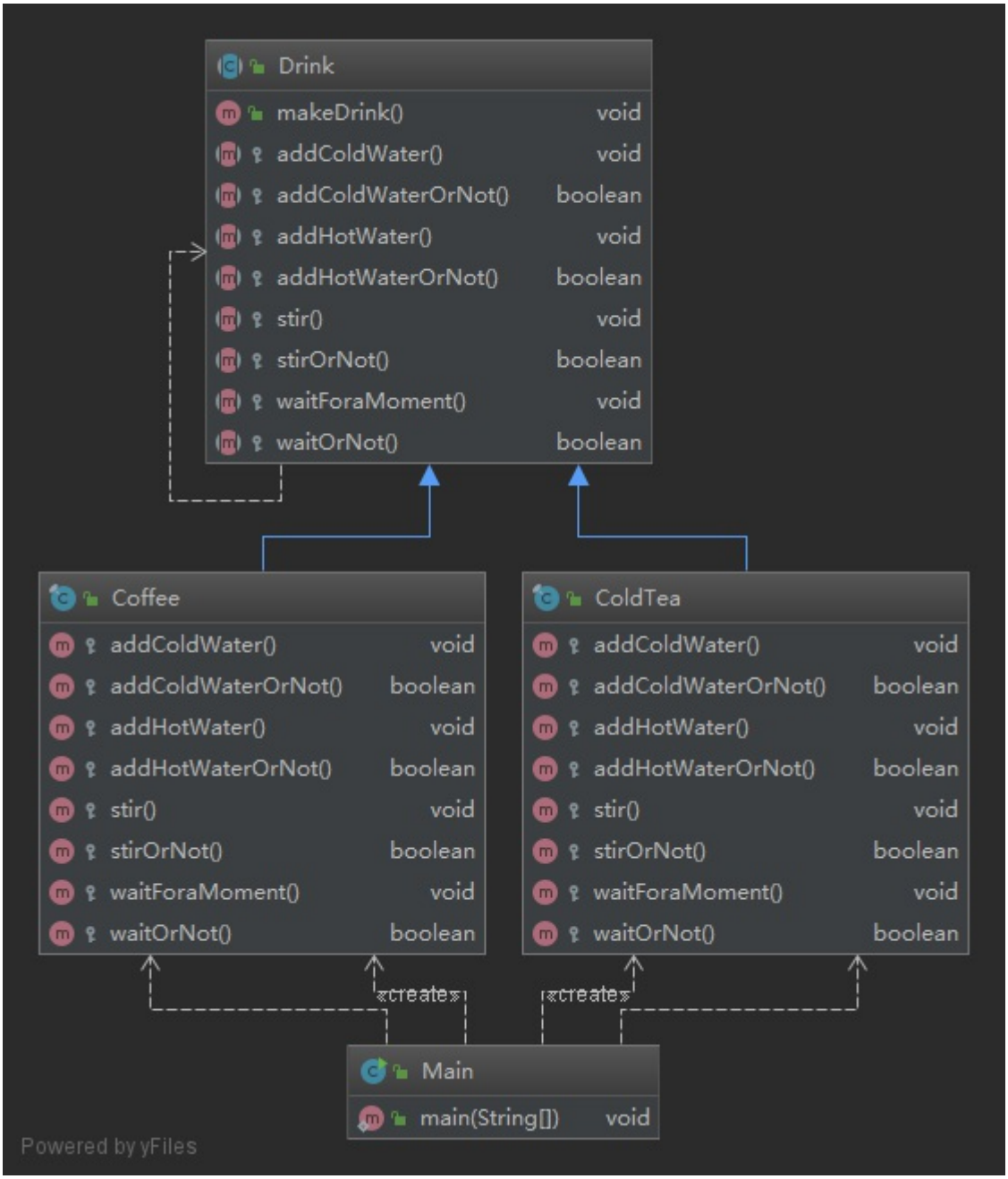
问题描述

模拟泡饮料的过程

1. 加水
2. 搅拌
3. 等一会, 等饮料凉一会儿

将具体的功能和hook函数下放给子类

UML 类图



Code

Drink.java

```
package Template;

public abstract class Drink {
    public void makeDrink() {
        if (addColdWaterOrNot()) addColdWater();
        if (addHotWaterOrNot()) addHotWater();
        if (stirOrNot()) stir();
        if (waitOrNot()) waitForaMoment();
    }

    protected abstract void addColdWater();

    protected abstract boolean addColdWaterOrNot();

    protected abstract void addHotWater();
```

```
    protected abstract boolean addHotWaterOrNot();

    protected abstract void stir();

    protected abstract boolean stirOrNot();

    protected abstract void waitForaMoment();

    protected abstract boolean waitOrNot();
}
```

Coffee.java

```
package Template;

public final class Coffee extends Drink {
    @Override
    protected void addColdWater() {
    }

    @Override
    protected boolean addColdWaterOrNot() {
        return false;
    }

    @Override
    protected void addHotWater() {
        System.out.println("add hot water.");
    }

    @Override
    protected boolean addHotWaterOrNot() {
        return true;
    }

    @Override
    protected void stir() {
        System.out.println("stir.");
    }

    @Override
    protected boolean stirOrNot() {
        return true;
    }

    @Override
    protected void waitForaMoment() {
        System.out.println("wait....");
    }

    @Override
    protected boolean waitOrNot() {
        return true;
    }
}
```

ColdTea.java

```
package Template;

public final class ColdTea extends Drink {
    @Override
    protected void addColdWater() {

    }

    @Override
    protected boolean addColdWaterOrNot() {
        return false;
    }

    @Override
    protected void addHotWater() {
        System.out.println("add hot water, but not too hot.");
    }
```

```
    @Override
    protected boolean addHotWaterOrNot() {
        return true;
    }

    @Override
    protected void stir() {

    }

    @Override
    protected boolean stirOrNot() {
        return false;
    }

    @Override
    protected void waitForaMoment() {
        System.out.println("wait for a long time.");
    }

    @Override
    protected boolean waitOrNot() {
        return true;
    }
}
```

运行结果

Main.java

```
package Template;

public class Main {
    public static void main(String[] args) {
        Coffee coffee = new Coffee();
        System.out.println("make coffee.");
        coffee.makeDrink();
        System.out.println("****************");
        ColdTea coldTea = new ColdTea();
        System.out.println("make cold tea.");
        coldTea.makeDrink();
    }
}
```

out

```
make coffee.
add hot water.
stir.
wait....
****************
make cold tea.
add hot water, but not too hot.
wait for a long time.
```