14. 枚举

14.1. 认识枚举

- 枚举就是要让某个类型的变量的取值只能为若干个固定值中的一个否则编译器就会报错，枚举可以让编译器在编译时就可以控制源程序赋给的非法值，使用普通变量的方式在开发阶段无法实现这一目标
- 在JDK1.5之后，使用关键字enum定义的一种新类型，称为枚举类型

14.2 Enum类与enum关键字

- 使用enum关键字定义的枚举类，实际上就相当于定义了一个类，此类继承了Enum类而已
- Enum类中定义了如下的方法：

protected Enum(String name, int ordinal)	此构造方法不能被外部直接调用，只能被其子类访问，此构造方法为自动调用
public final String name()	枚举的名字
public final int ordinal()	枚举

14.3. 集合对枚举的支持

- 在JDK1.5之后，对于Set和Map接口而言又增加了两个新的子类：EnumSet、EnumMap两个类

14.4. 带构造方法的枚举

```
/**
 * @author xiao儿
 * @date 2019/9/4 8:24
 * @Description Color
 * <p>
 * 定义一个枚举类
 */
public enum Color {
    RED(10), GREEN(20), BLUE();

    private int color;

```

```
    private Color() {
        System.out.println("无参数的构造方法");
    }

    Color(int color) {
        this.color = color;
        System.out.println("有参数的构造方法");
    }
}
```

14.5. 让枚举实现接口

```
public enum Color implements Info {
    RED(10), GREEN(20), BLUE();

    private int color;

    private Color() {
        System.out.println("无参数的构造方法");
    }

    Color(int color) {
        this.color = color;
        System.out.println("有参数的构造方法");
    }

    @Override
    public int getColor() {
        return color;
    }
}
```

14.6. 在枚举中定义抽象方法

```
public enum Color implements Info {
    RED(10) {
        @Override
        public String getColor2() {
            return "红色";
        }
    }, GREEN(20) {
        @Override
        public String getColor2() {
            return "绿色";
        }
    }, BLUE {
        @Override
        public String getColor2() {
            return "蓝色";
        }
    };

    private int color;

    private Color() {
        System.out.println("无参数的构造方法");
    }
```

```
24
25      Color(int color) {
26          this.color = color;
27          System.out.println("有参数的构造方法");
28      }
29
30      @Override
31      public int getColor() {
32          return color;
33      }
34
35      public abstract String getColor2();
36  }
```

####