Java 核心技术

第五章 继承、接口和抽象类

第二节 抽象类和接口

华东师范大学 陈良育

抽象类(1)

- 类：属性(0或多个)+方法(0或多个)
- 一个完整(健康)的类：所有的方法都有实现(方法体)
- 类可以没有方法，但是有方法就肯定要有实现，这才是一个完整的类
- 一个完整的类才可以被实例化，被new出来
- 如果一个类暂时有方法未实现，需要被定义为抽象类

抽象类(2)

```
public abstract class Shape {
    int area;
    public abstract void calArea();
}
```

- 抽象类关键字abstract声明
- 抽象类的组成
  - (optional)成员变量，个数不限
  - (optional)具体方法，方法有实现，个数不限
  - (optional)抽象方法，加abstract关键字，个数不限

抽象类(3)

- 抽象类也是类。一个类继承于抽象类，就不能继承于其他的(抽象)类.
- 子类可以继承于抽象类，但是一定要实现父类们所有abstract的方法。如果不能完全实现，那么子类也必须被定义为抽象类。
- 只有实现父类(们)的所有抽象方法，才变成完整类。
- 参看Shape和Rectangle的例子

接口(1)

- 如果类的所有方法都没有实现，那么这个类就算是接口interface。

```
public interface Animal {
    public void eat();
    public void move();
}
```

- 接口不算类，或者说是“特殊”的类。
- 类只可以继承(extends)一个类，但是可以实现(implements)多个接口，继承和实现可以同时。

接口(2)

- 接口可以继承(多个)接口，没有实现的方法将会叠加
- 类实现接口，就必须实现所有未实现的方法。如果没有全部实现，那么只能成为一个抽象类。
- 接口里可以定义变量，但是一般是常量，详细可以参考final节。
- 例子Animal/Cat
- 例子Animal/ClimbTree/LandAnimal/Rabbit
- 例子Animal/ClimbTree/CatFamily/Tiger

总结

- 抽象类和接口相同点：两者都不能被实例化，不能new操作
- 抽象类和接口不同点
  - 抽象类abstract, 接口interface
  - 抽象类可以有部分方法实现，接口所有方法不能有实现
  - 一个类只能继承(extends)一个(抽象)类，实现(implements)多个接口
  - 接口可以继承(extends)多个接口
  - 抽象类有构造函数，接口没有构造函数
  - 抽象类可以有main，也能运行，接口没有main函数
  - 抽象类方法可以有private/protected, 接口方法都是public

代码(1) Shape/Rectangle.java


```java
public abstract class Shape {
    //面积
    int area;

    //计算面积方法
    public abstract void calArea();
}
```

```java
//继承自Shape抽象类
public class Rectangle extends Shape{

    int width;  //宽
    int length; //长

    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }

    public void calArea() {
        System.out.println(this.length * this.width);
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle(10,5);
        rect.calArea();
    }
}
```

代码(2) Animal/Cat/ClimbTree.java


```java
public interface Animal {
    public void eat();
    public void move();
}
```

```java
public interface ClimbTree {
    public void climb();
}
```

```java
public class Cat implements Animal
{
    public void eat() {
        System.out.println("Cat: I can eat");
    }

    public void move(){
        System.out.println("Cat: I can move");
    }
}
```

代码(3) LandAnimal/Rabbit.java


```
public abstract class LandAnimal implements Animal {

    public abstract void eat() ;

    public void move() {
        System.out.println("I can walk by feet");
    }
}
```

```
public class Rabbit extends LandAnimal implements ClimbTree {

    public void climb() {
        System.out.println("Rabbit: I can climb");
    }

    public void eat() {
        System.out.println("Rabbit: I can eat");
    }
}
```

代码(4) CatFamily/Tiger.java


```
public interface CatFamily extends Animal, ClimbTree{
    //包含以下三个方法
    //eat()
    //move()
    //climb()
}

public class Tiger implements CatFamily {
    //必须实现CatFamily中的三个方法
    public void eat() {
        System.out.println("Tiger: I can eat");
    }

    public void move() {
        System.out.println("Tiger: I can move");
    }

    public void climb() {
        System.out.println("Tiger: I can climb");
    }
}
```

谢 谢！