多态

武永亮

讲授思路

- 多态的概念
- 多态的实现

多态的概念

- 多态：发送消息给某个对象，让该对象自行决定响应何种行为。
- 多态是通过方法的重载、重写实现的，要了解Java中的多态必须先了解“向上转型”。
  - 定义了一个子类Teacher，继承自Person。
    - 通过Teacher teacher = new Teacher()实例化Teacher对象；
    - 通过Person p= new Teacher();表示定义了一个Person类型的引用，指向新建的Teacher类型的对象，这就称为”向上转型”；
  - “向上转型”既可以使用子类强大的功能，又可以抽取父类的共性。

多态的实现

- 通过将子类对象引用赋值给超类对象引用变量来实现动态方法调用。

```
class Person{
    private String name;
    public  void display() {
        System.out.println("Person display");
    }
}
class Teacher extends Person
    public void display() {
        System.out.println("Tea
    }
    public void displayEx {
        System.out.println("Ext
    }
}
```

```
public class Test{
    public static void main(String[] args){
        Person person = new Teacher(); //向上转型
        person.display();
        person.displayEx();
    }
}
```

多态的实现

- 思考：为什么子类的类型的对象实例可以赋给父类引用？
  - 自动实现向上转型。通过Person person = new Teacher();语句，编译器自动将子类实例向上移动，成为通用类型Person。
- 思考： person.display();将执行子类还是父类定义的方法？
  - 子类的。在运行时期，将根据person这个对象引用实际的类型来获取对应的方法。所以才有多态性。一个基类的对象引用，被赋予不同的子类对象引用，执行该方法时，将表现出不同的行为。

```
public class Test{
    public static void main(String[] args){
        Person person = new Teacher(); //向上转型
        person.display();
        person.displayEx();
    }
}
```

多态的实现

- 对于父类中定义的方法，如果子类中重写了该方法，那么父类类型的引用将会调用子类中的定义的这个方法，这就是动态连接。
- 父类中的一个方法只有在父类中定义而在子类中没有重写的情况下，才可以被父类类型的引用调用。
- 对于子类中定义而父类中没有的方法，它是无可奈何的。

多态的实现

```
class Person {
        private String name;
        public  void display() {
            System.out.println("Person display");
        }
}
class Teacher extends Person {
        public void display() {
            System.out.println("Teacher display");
        }
}
class Student extends Person{
        public void display(){
            System.out.println("Student display");
        }
        public void displayEx{
            System.out.println("Extend from Person");
        }
}
```

```
public class Test{
    public static void main(String[] args){
        Person p = new Teacher();
        p.display();
        p = new Student();
        p.display();

    }
}
```

```
Teacher display
Student display
```

总结

- 多态的概念
- 多态的实现

Thank You