面向对象第四天：

潜艇游戏第一天：

潜艇游戏第二天：

潜艇游戏第三天：

1. 创建侦察潜艇数组、鱼雷潜艇数组、水雷潜艇数组，水雷数组，炸弹数组，并测试
2. 设计SeaObject超类，6个类继承超类
3. 在SeaObject中设计两个构造方法，6个派生类分别调用

潜艇游戏第四天：

1. 将侦察潜艇数组、鱼雷潜艇数组、水雷潜艇数组统一组合为SeaObject数组，并测试
2. 在6个类中重写move()移动，并测试（单元测试：指对软件中的最小可测试单元进行检查和验证，如C语言中单元指一个函数，Java中单元指一个类）
3. 画窗口：在World类中（不要求掌握，复制粘贴即可）
    - import JFrame+JPanel
    - 设计World类继承JPanel
    - main中代码复制粘贴

笔记：

1. 向上造型：（好处：代码复用）
    - 超类型的引用指向派生类的对象
    - 能点出来什么，看引用的类型

    > 多种角色能干的事都一样时，可以将多种角色统一向上造型到超类数组中，实现代码复用
    >
    > eg: 学生/老师/医生都是输出名字+问好干的事都一样，就可以将学生/老师/医生统一向上造型到Person数组中，这样用一个for即可全部输出名字+问好（代码复用）

```
public class UploadDemo {
    public static void main(String[] args) {
        Aoo o1 = new Aoo();
        o1.a = 1;
        o1.show();
        //o1.b = 2; //编译错误，超类不能访问派生类特有的
        //o1.test();

        Boo o2 = new Boo();
        o2.b = 1;
        o2.test();
        o2.a = 2; //正确，派生类可以访问超类的
        o2.show();

        Aoo o3 = new Boo(); //向上造型
        o3.a = 1;
```

```
        o3.show();
        //o3.b = 2; //编译错误，能点出来什么，要看引用的类型
        //o3.test();
    }
}

public class Aoo {
    int a;
    void show() {}
}

public class Boo extends Aoo {
    int b;
    void test() {}
}
```

2. 方法的重写(override/overriding)：重新写一个方法
    - 发生在父子类中，方法名相同，参数列表相同
    - 重写方法被调用时，看new的对象是什么类型

```
public class test123 {
    class 餐馆 {
        void 做餐() {
            做中餐
        }
    }

    //1)我还是想做中餐：不需要重写
    class Aoo extends 餐馆 {
    }

    //2)我想改做西餐：需要重写
    class Aoo extends 餐馆 {
        void 做餐() {
            做西餐
        }
    }

    //3)我想在中餐基础之上加入西餐：需要重写(先super中餐，再加入西餐)
    classs Aoo extends 餐馆 {
        void 做餐() {
            super.做餐()
            做西餐
        }
    }
}
```

    - 重写需遵循"两同两小一大"原则：（了解即可，一般都是一样的）
        - 两同：
            - 方法名相同
            - 参数列表相同
        - 两小：
            - 派生类方法的返回值类型小于或等于超类方法的
                - void和基本类型时，必须相等

- 引用类型时，小于或等于（派生类小，超类大）

```
public class test123 {
    //超类大，派生类小
    class Coo {
        void show() {}
        double test() { return 0.0; }
        Student say() { return null; }
        Person sayHi() { return null; }
    }

    class Doo extends Coo {
        //int show() { return 1; } //编译错误，void时必须相等
        //int test() { return 0; } //编译错误，基本类型时必须相等
        //Person say() { return null; } //编译错误，引用类型时必须小于或等于
        Student sayHi() { return null; }
    }
}
```

  - 派生类方法抛出的异常小于或等于超类方法的
- 一大：
  - 派生类方法的访问权限大于或等于超类方法的

3. 重写与重载的区别：
   - 重写(override)：发生在父子类中，方法名相同，参数列表相同
   - 重载(overload)：发生在同一类中，方法名相同，参数列表不同

作业：

```
/*
作业
写在ooday05包中
如下的类必须分在不同的文件中写
1.创建Person类，包含：
  1)成员变量:name,age,address
  2)构造方法:Person(3个参数){ 赋值 }
  3)方法:sayHi(){ 输出3个数据 }
2.创建学生类Student，继承Person，包含：
  1)成员变量:学号stuId(String)
  2)构造方法:Student(4个参数){ super调超类3参构造、赋值stuId }
  3)方法:重写sayHi(){ 输出4个数据 }
3.创建老师类Teacher，继承Person，包含：
  1)成员变量:工资salary(double)
  2)构造方法:Teacher(4个参数){ super调超类3参构造、赋值salary }
  3)方法:重写sayHi(){ 输出4个数据 }
4.创建医生类Doctor，继承Person，包含：
  1)成员变量:职称level(String)
  2)构造方法:Doctor(4个参数){ super调超类3参构造、赋值level }
5.创建测试类Test，main中：
  1)创建Person数组ps,包含5个元素,给元素赋值(学生/老师/医生),遍历输出名字并问好
 */
```

```
//Person.java
public class Person {
```

```
    String name;
    int age;
    String address;

    Person(String name, int age, String address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }

    void sayHi() {
        System.out.println("我的名字是" + name + "，年龄是" + age + "，地址是" +
address);
    }
}
```

```
//Student.java
public class Student extends Person {
    String stuID;

    Student(String name, int age, String address, String stuID) {
        super(name, age, address);
        this.stuID = stuID;
    }

    @Override
    void sayHi() {
        System.out.println("我的名字是" + name + "，年龄是" + age + "，地址是" +
address + "，学号是" + stuID);
    }
}
```

```
//Teacher.java
public class Teacher extends Person {
    double salary;

    Teacher(String name, int age, String address, double salary) {
        super(name, age, address);
        this.salary = salary;
    }

    @Override
    void sayHi() {
        System.out.println("我的名字是" + name + "，年龄是" + age + "，地址是" +
address + "，工资是" + salary);
    }
}
```

```java
//Doctor.java
public class Doctor extends Person {
    String level;

    Doctor(String name, int age, String address, String level) {
        super(name, age, address);
        this.level = level;
    }
}
```

```java
//Test.java
public class Test {
    public static void main(String[] args) {
        Person[] ps = new Person[5];
        ps[0] = new Student("张三", 21, "广州", "20220101"); //向上造型
        ps[1] = new Student("李四", 22, "深圳", "20220102");
        ps[2] = new Teacher("王五", 23, "佛山", 5000.0);
        ps[3] = new Teacher("赵六", 24, "东莞", 6000.0);
        ps[4] = new Doctor("孙七", 25, "中山", "正高");

        for (int i = 0; i < ps.length; i++) {
            Person p = ps[i];
            System.out.println(p.name);
            p.sayHi();
        }

        Person p = new Student("zhangsan", 25, "LF", "111");
        p.sayHi();
        //1)p是Person类型的，所以能点出Person中的所有东西，包括了sayHi()
        //2)p的对象是Student类型的，重写方法被调用要看对象，所以调用的是Student类的
sayHi()
    }
}
```

补充：

1. 超类的意义：
    - 封装共有的属性和行为（实现代码复用）
    - 为所有派生类提供统一的类型，即向上造型（实现代码复用）
2. is-a：

```java
Animal o = new Animal(); //动物是动物
Tiger o = new Tiger(); //老虎是老虎
Animal o = new Tiger(); //老虎是动物：语义通
Tiger o = new Animal(); //动物是老虎：语义不通，编译错误
class Animal { //动物类
}
class Tiger extends Animal { //老虎类
}
```

3. 明日单词：

```
1)override:重写
   2)package:包
```

```
3)import:导入
4)public:公开的
5)protected:受保护的
6)private:私有的
7)card:卡
8)id:号码
9)password/pwd:密码
10)balance:余额
11)pay:支付
12)money:金额
13)check:检查
14)static:静态的
15)image:图片
16)icon:图标
17)get:获取
18)status:状态
```

晚课:

1. 重写与重载:

```
class Aoo {
    void show() {}
}
class Boo {
    void show() {}
}
//既没有重写，也没有重载

class Aoo {
    void show() {}
}
class Boo extends Aoo {
    void show() {}
}
//发生了重写

class Aoo {
    void show() {}
}
class Boo {
    void show(int a) {}
}
//既没有重写，也没有重载

class Aoo {
    void show() {}
}
class Boo extends Aoo {
    void show(int a) {}
}
//此类发生了show的重载
//继承了Aoo，已经有了Aoo的void show()方法，自己再写一个void show(int a)方法，相当于重载
```

2. 继承要符合is(是)的关系，不能为了复用代码就乱继承，类与类之间的关系有很多种(继承、关联、组合、聚合...)
3. 继承意味着代码虽然我没有写，但也属于我，只是没有写在一起而已
4. 继承的是超类中的成员变量和普通方法，不包括构造方法

   超类的构造方法是被派生类通过super来调用的