面向对象第6天：

潜艇游戏第六天：

1. 设计窗口的宽和高为常量，适当地方做修改
2. 画对象

```
1)想画对象需要获取对象的图片，每个对象都要获取图片，
  意味着获取图片行为为共有行为，所以设计在SeaObject超类中，
  每个对象获取图片的代码都是不一样的，所以设计为抽象方法
  ----在SeaObject中设计为抽象方法getImage()获取对象的图片
2)在派生类中重写getImage()获取对象图片
  ----在6个类中重写getImage()返回不同的图片
3)因为只有活着的对象才需要画到窗口中，所以需要设计对象的状态(活着还是死了)，
  每个对象都有状态，意味着状态为共有属性，所以设计在SeaObject超类中，
  状态一般都设计为常量，同时再设计state变量表示当前状态
  ----在SeaObject中设计LIVE、DEAD常量，state变量表示当前状态
  在后期的业务中经常需要判断对象的状态，每个对象都得判断，
  意味着判断状态的行为为共有行为，所以设计在SeaObject超类中，
  每个对象判断状态的代码都是一样的，所以设计为普通方法
  ----在SeaObject中设计isLive()、isDead()判断对象的状态
4)数据(状态、图片、x坐标、y坐标)都有了就可以开画了，每个对象都得画，
  意味着画对象行为为共有行为，所以设计在SeaObject超类中，
  每个对象画的代码都是一样的，所以设计为普通方法
  ----在SeaObject中设计paintImage()画对象
5)画对象的行为做好了，在窗口World类中调用即可
  5.1)准备对象
  5.2)重写paint()方法----在paint()中调用paintImage()画对象即可
```

笔记：

1. static final常量：应用率高
   - 必须声明的同时初始化
   - 常常由(类名.)来访问，不能被改变
   - 建议：常量所有字母都大写，多个单词用下划线_分隔
   - 编译器在编译时会将常量直接替换为具体的值，效率高
   - 何时用：数据永远不变，并且经常使用

```
public class StaticFinalDemo {
    public static void main(String[] args) {
        System.out.println(Aoo.PI); //常常通过(类名.)来访问
        //Aoo.PI = 3.1415926; //编译错误，常量不能被改变

        //1)加载Boo.class到方法区中
        //2)静态变量num一并存储在方法区中
        //3)到方法区中获取num的值并输出
        System.out.println(Boo.num);

        //编译器在编译时会将常量直接替换为具体的值，效率高
        //相当于System.out.println(5);
```

```
        System.out.println(Boo.COUNT);
    }
}

class Boo {
    public static int num = 5; //静态变量
    public static final int COUNT = 5; //常量
}

class Aoo {
    public static final double PI = 3.14159;
    //public static final int NUM; //编译错误，常量必须在声明的同时初始化
}
```

2. 抽象方法：
    - 由abstract修饰
    - 只有方法的定义，没有具体的实现(连大括号{}都没有)
3. 抽象类：
    - 由abstract修饰
    - 包含抽象方法的类必须是抽象类
    - 抽象类不能被实例化
    - 抽象类是需要被继承的，派生类：
        - 重写抽象方法(变不完整为完整)----一般做法
        - 也声明为抽象类----------一般不这么做
    - 抽象类的意义：（也包含了超类的意义）
        - 封装共有的属性和行为---------代码复用
        - 为所有派生类提供统一的类型----向上造型(代码复用)
        - 可以包含抽象方法，为所有派生类提供统一的入口(造型之后能点出来)，同时可以达到强制必须重写的目的(相当于制定一个规则)

补充：

1. 设计规则：
    - 将共有的属性和行为，抽到超类中（抽共性）
    - 若派生类的行为都一样，设计为普通方法

      若派生类的行为都不一样，设计为抽象方法
    - 。。。
2. 抽象方法/抽象类的疑问：
    - 抽象方法存在的意义是什么？
        - 保证当发生向上造型时，通过超类的引用能点出来那个方法（保证能点出方法来）
    - 既然抽象方法的意义是保证能点出来，那为什么不设计为普通方法呢？
        - 设计为普通方法，意味着派生类可以重写也可以不重写，但设计为抽象方法，可以强制派生类必须重写（强制派生类重写的目的）
3. 画对象带数：
    - 当创建World对象时，会分配World类中的成员变量（创建对象）
    - frame.setVisible(true)时会自动调用paint()方法，在paint()中：当第1次用到Images类时，会将Images.class加载到方法区中，同时将静态图片分配到方法区中，同时自动执行静态块给静态图片赋值

- 假设战舰调用paintImage()，意味着this指的就是战舰，方法中判断战舰是否是活着的，若活着则获取战舰的图片，画到战舰的x坐标和y坐标上

4. 明日单词：

```
1)inner:内部的
2)outer:外部的
3)baby:孩子
4)create:创建
5)anonymous/anon:匿名
6)shoot:射击
7)next:下
8)one:一个
9)action:行动
10)enter:进入
11)timer:定时器
12)interval:间隔
13)schedule:日程表
14)task:任务
15)repaint:重新画
```