面向对象第7天：

潜艇游戏第七天：

1. 潜艇入场：
   - 潜艇是由窗口产生的，所以在窗口World类中设计nextSubmarine()生成潜艇对象
   - 潜艇入场为定时发生的，所以在run()中调用submarineEnterAction()实现潜艇入场

     在submarineEnterAction()中：

     每400毫秒，获取潜艇对象obj，submarines扩容，将obj添加到最后一个元素上

     > 注意：在run()中调用submarineEnterAction()之后，一定要调用repaint()来重画
2. 水雷入场：（前半段）
   - 水雷是由水雷潜艇发射出来的，所以在MineSubmarine中设计shootMine()生成水雷对象
   - 水雷入场为定时发生的，所以在run()中调用mineEnterAction()实现水雷入场

     在mineEnterAction()中：

     每1000毫秒，.....
3. 海洋对象移动：
   - 海洋对象移动为共有行为，所以在SeaObject中设计抽象方法move()实现移动，派生类中重写
   - 海洋对象移动为定时发生的，所以在run()中调用moveAction()实现海洋对象移动

     在moveAction()中：

     遍历所有潜艇：潜艇动，遍历所有水雷：水雷动，遍历所有炸弹：炸弹动

笔记：

1. 成员内部类：（了解即可，应用率不高）
   - 类中套类，外面的称为外部类，里面的称为内部类
   - 内部类只服务于外部类，对外不具备可见性
   - 内部类对象通常在外部类中创建（实例变量）
   - 内部类中可以直接访问外部类的成员（变量、方法）（包括私有的），

     内部类中有一个隐式的引用指向创建它的外部类对象（外部类名.this）

```
//成员内部类的演示
public class InnerClassDemo {
    public static void main(String[] args) {
        Mama m = new Mama();
        //Baby b = new Baby(); //编译错误，内部类对外不具备可见性
        m.create().show();
    }
}

class Mama { //外部类
    private String name;

    Baby create() {
```

```
        Baby b = new Baby(); //正确，内部类对象通常在外部类中创建
        return b;
    }

    class Baby { //成员内部类
        void show() {
            System.out.println(name); //简写
            System.out.println(Mama.this.name); //完整写法
            //System.out.println(this.name); //编译错误，当前Baby类没有name
属性
        }
    }
}
```

2. 匿名内部类：（应用率高）
   - 何时用：若想创建一个类(派生类)的对象，并且对象只被创建一次，可以设计为匿名内部类，可以大大简化代码操作
   - 在匿名内部类中不能修改外面局部变量的值，因为在匿名内部类中外面局部变量会被默认为final
   - 小面试题：
     - 问：内部类有独立的.class字节码文件吗?
     - 答：有（在项目的out文件夹中）

```
//匿名内部类的演示
public class NstInnerClassDemo {
    public static void main(String[] args) {
        //1)创建了Aoo的一个派生类，但是没有名字
        //2)为该派生类创建了一个对象，名为o1----向上造型为Aoo类型
        //  ---new Aoo(){}是在创建Aoo的派生类的对象
        //3)大括号中的为派生类的类体
        Aoo o1 = new Aoo() {

        };

        int num = 5;
        //1)创建了Boo的一个派生类，但是没有名字
        //2)为该派生类创建了一个对象，名为o3----向上造型为Boo类型
        //3)大括号中的为派生类的类体
        Boo o3 = new Boo() {
            void show() { //重写
                System.out.println("show");
                //num = 55; //编译错误，在此处会默认外面局部变量num为final
            }
        };
        o3.show();
    }
}

abstract class Boo {
    abstract void show();
}

abstract class Aoo {
}
```

补充：

1. 隐式引用：
    - this：指代当前对象
    - super：指代当前对象的超类对象
    - 外部类名.this：指代当前对象的外部类对象
2. 做功能的套路：（重要！）
    - 先写行为/功能/方法：
        - 若为某个对象所特有的行为/功能，就将方法设计在特定的类中
        - 若为对象所共有的行为/功能，就将方法设计在超类中
    - 窗口调用：
        - 若为定时发生的，就在定时器中调用
        - 若为事件触发的，就在侦听器中调用
3. 如何调错：（不能着急，得慢慢调）
    - 打桩：System.out.println(数据);
4. 明日单词：

```
1)interface:接口
2)implements:实现
3)enemy:敌人
4)nuclear:核武器
5)left:左
6)right:右
7)out of bounds:超出界限
8)Key:键盘
9)Adapter:适配器
10)release:松开/弹起
11)code:编码
12)space:空白
13)listener:监听
```