面向对象第九天：

潜艇游戏第九天：

1. 水雷入场：（后半段）
   - 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，遍历所有潜艇，判断若为水雷潜艇，则强转为水雷潜艇类型，

     获取水雷对象obj，mines扩容，将obj装到最后一个元素上
2. 炸弹与潜艇的碰撞：
   - 在SeaObject中设计isHit()检测碰撞、goDead()去死。在Battleship中设计addLife()增命

```
class SeaObject { //检测碰撞
    public boolean isHit(SeaObject other) {
        //this:当前对象
        //other:另一个对象
    }
}

假设：s表示潜艇，b表示炸弹，ship表示战舰，m表示水雷
（两个对象之间碰撞检测代码都是一样的）
1)s.isHit(b);    //this:潜艇  other:炸弹
2)b.isHit(s);    //this:炸弹  other:潜艇
3)ship.isHit(m); //this:战舰  other:水雷
4)m.isHit(ship); //this:水雷  other:战舰
```

   - 炸弹与潜艇的碰撞为定时发生的（炸弹发射出去后，与潜艇的碰撞与否是定死的），所以在run()中设计bombBangAction()实现炸弹与潜艇碰撞

     在bombBangAction()中：

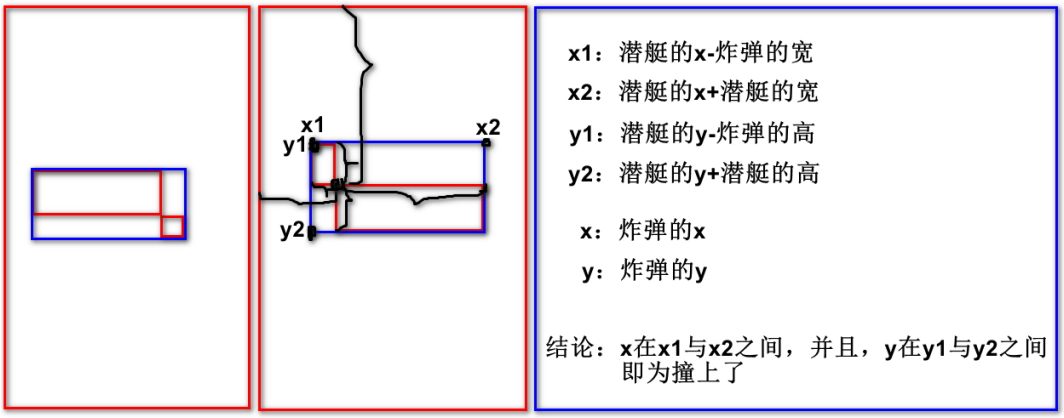
     遍历所有炸弹得炸弹，遍历所有潜艇得潜艇，判断若都活着并且还撞上了：

     炸弹去死、潜艇去死

     判断若是分，则强转为得分接口，玩家得分

     判断若是命，则强转为得命接口，获取命数，战舰得命

- 碰撞检测图



3. 画分和画命：
   - 在Battleship中设计getLife()获取命数
   - 在World的paint()中：画分和画命

笔记：

1. 多态：多种形态
   - 意义：
     - 同一个对象被造型为不同的类型时，有不同的功能（所有对象都是多态的）

       对象多态：我、你、水.....

```
我 me = new 我();
讲师 o1 = me;
孩子他妈 o2 = me;
老公的老婆 o3 = me;
o1.授课();
o2.揍他();
o3.收工资();

interface 讲师 {
    void 授课();
}

interface 孩子他妈 {
    void 揍他();
}

interface 老公的老婆 {
    void 收工资();
}

class 我 implements 讲师, 孩子他妈, 老公的老婆 {
    public void 授课() {
    }

    public void 揍他() {
    }

    public void 收工资() {
```

```
    }
}
```

- 同一类型的引用指向不同的对象时，有不同的实现（所有抽象方法都是多态的）

  行为多态：cut()、getImage()、move()、getScore().....

```
人 o1 = new 理发师();
人 o2 = new 厨师();
人 o3 = new 外科医生();
o1.cut(); //剪发
o2.cut(); //切菜
o3.cut(); //开刀

abstract class 人 {
    abstract void cut();
}

class 理发师 extends 人 {
    void cut() {
        剪发
    }
}

class 厨师 extends 人 {
    void cut() {
        切菜
    }
}

class 外科医生 extends 人 {
    void cut() {
        开刀
    }
}
```

- 向上造型/自动类型转换：
  - 超类型的引用指向派生类的对象(前面是超类型，后面是派生类型)
  - 能点出来什么，看引用的类型
  - 能向上造型成为的类型有：超类，所实现的接口
- 强制类型转换，成功的条件只有如下两种：
  - 引用所指向的对象，就是该类型
  - 引用所指向的对象，实现了该接口或继承了该类
- 强转若不符合如上条件，则发生ClassCastException类型转换异常

  建议：在强转之前一定要先通过instanceof来判断引用的对象是否是该类型

> 强转时若符合如上的两个条件，则instanceof返回true，若不符合则返回false
>
> 何时需要强转：你想访问的东西在超类中没有，那就需要强转

```
public class MultiType {
    public static void main(String[] args) {
        //条件1:引用所指向的对象，就是该类型
        //条件2:引用所指向的对象，实现了该接口或继承了该类
        Aoo o = new Boo(); //向上造型
        Boo o1 = (Boo) o; //引用o所指向的对象，就是Boo类型（符合条件1）
```

```
        Inter o2 = (Inter) o; //引用o所指向的对象，实现了Inter接口（符合条件2）
        //Coo o3 = (Coo)o; //运行时会发生ClassCastException类型转换异常

        //判断o是否是Coo类型(与强转成功条件完全匹配)
        if (o instanceof Coo) { //false
            Coo o4 = (Coo) o; //instanceof若为true，则强转一定成功
        } else {
            System.out.println("o不是Coo类型");
        }
    }
}

interface Inter {
}

class Aoo {
}

class Boo extends Aoo implements Inter {
}

class Coo extends Aoo {
}
```

补充：

1. 体会接口的好处：

```
//复用性好、扩展性好、维护性好（高质量代码）
if (s instanceof EnemyScore) { //适用于所有实现EnemyScore接口
    EnemyScore es = (EnemyScore) s;
    score += es.getScore();
}

if (s instanceof EnemyLife) { //适用于所有实现EnemyLife接口
    EnemyLife el = (EnemyLife) s;
    int num = el.getLife();
    ship.addLife(num);
}
```

2. 明日单词：

```
1)subtract:减
2)gameover:结束
3)running:运行
```