实验一

实验描述：

Java中的String类创建出来的字符串内容是不可变的，来自己编写一个字符串类MyString来实现类似StringBuilder的功能，实现如下的字符串相关操作。

实验要求：

1.一个成员变量char)mystring来表示字符串
2.实现如下的方法：

- 无参构造方法public MyString(0。注意无参构造方法中需要对成员变量mystring初始化，避免出现NullPointerException
- 有参构造方法public MyString(char[Uch)。初始化成员变量mystring
- 显示当前这个字符串public char[]showContent),返回当前这个字符串，与print配合使用可以显示出当前字符串
- 获取指定字符第一次出现的下标public int indexOf(char ch)。注意对字符串中没有指定字符的处理
- 获取指定下标的字符public char charAt(int index)。注意对非法下标的处理，可以返回特定字符表示输入的是非法字符。index的取值应在[0,string.length]
- 获取字符串的子串public MyString substring(int begin,int end)
- 向字符串末尾添加新的字符串public MyString append(char[]ch)。实现对当前字符串末尾就地添加新字符串的功能

核心代码展示：

```
    public String substring(int begin,int end){
        String s=new String(mystring,begin,end);
        return s;
    }
```

```
    public  void append(char[] ch){
        char[] temp;
        temp=new char[mystring.length+ch.length];
        System.arraycopy(mystring,0,temp,0,mystring.length);
        System.arraycopy(ch,0,temp,mystring.length,ch.length);
        mystring=temp;
    }
```

成果展示：

```
请更新
请更新i
new string
获取w字符第一次出现下标：2
获取下标-1的字符：#
new string is here
获取h字符第一次出现下标：14
获取string2对象的0~3的子串：new
string3
获取h字符第一次出现下标：-1
```

实验二

实验要求：

项目由以下类组成：

(1) FootballPlayer class: 所有球员类的父类，用来存储球员的信息

(2) 根据球员位置而设定的一些特定的类，均为 FootballPlayer 的子类

(3) FootballTeam class：用来存储球队的信息，会包括所有球员的对象

(4) FootballTester class：创建对象并测试各个类中的方法

具体要求如下：

1. FootballPlayer class

(1) 包括如下实例变量：

(a) player name

(b) position: 在本项目中会涉及 quarterback, runningback 和 defensiveback

(c) team

(d) games played：本赛季参赛数目

(2) 构造方法：

(a) 默认构造方法：将所有实例变量设为默认值

(b) 有参构造方法：将所有实例变量设置为要设置的值

(3) 其他方法：

(a) playerRating 方法：计算球员的排名。在本类中直接打印一句话表明是 FootballPlayer 中的 playersRating 方法即可。将由子类具体实现。

(b) compareTo 方法：比较两个球员的排名高低。如果调用对象的排名比参数中的对象排名高，返回一个正数；如果调用对象的排名比参数中的对象排名低，返回一个负数；如果排名相同则返回 0。

(c) getter 和 setter 方法

(d) toString 方法：按照如下的格式返回球员对象的信息：

Name: Phil Sims, Position: Quarter Back, Team: Giants

2. Quarterback class:

继承自 FootballPlayer，专门面向 Quarterback 位置的球员

(1) 实例变量(all ints)：

(a) pass attempts

(b) pass completed

(c) touchdowns passing

(d) total yards passing

(2) 构造方法：

(a) 默认构造方法：将所有的实例变量设置为默认值。提示：也要传合适的参数给父类（超类）

(b) 有参构造方法

(3) 其他方法：

(a) double completionPercentage()：用来计算 pass 的成功率

公式：passes completed / pass attempts

(b) double averagePassingYardsPerGame()

公式：total yards passing / games played

(c) double averageTouchDownsPerGame()

公式：touchdowns passing / games played

(d) getter 和 setter 方法

(e) playerRating：重写超类中方法

公式：average touch downs per game + (completion percentage * 100) + (average passing yards per game / 5)

(e) toString 方法：按照如下的格式返回 Quarterback 球员对象的信息：（提示：利用超类的方法）

Name: Phil Sims, Position: Quarter Back, Team: Giants
Completion Percentage: 0.25, Average Passing Yards Per Game: 500.00
Average Touch Downs Per Game: 1.50, Player's Rating: 127

3. Runningback class:

   和 Quarterback class 类似，但有些许不同

   (1) 实例变量(all ints)：
      (a) running attempts
      (b) total running yards
      (c) touchdowns
   (2) 构造方法：要求同 Quarterback class
   (3) 其他方法：
      (a) 类似 Quarterback class 中的计算方法，返回值全部为 double，实现以下三个方法：
         averageYardsPerGame
         averageYardsPerAttempt
         averageTouchDownsPerGame
      (b) playerRating：重写超类中方法
         公式：averageTouchDownsPerGame() + averageYardsPerAttempt() + (averageYardsPerGame()/5)
      (c) getter 和 setter 方法
      (d) toString 方法：和 Quarterback 中类似，但要返回 Runningback 球员的信息

4. Defensiveback class:

   (1) 实例变量(all ints)：
      (a) tackles
      (b) interceptions
      (c) forced fumbles
   (2) 构造方法：要求同 Quarterback class
   (3) 其他方法：
      (a) 类似 Quarterback class 中的计算方法，返回值全部为 double，实现以下三个方法：
         averageTacklesPerGame
         averageInterceptionsPerGame
         averageForcedFumblesPerGame
      (b) playerRating：重写超类中方法
         公式：(averageTacklesPerGame() + averageInterceptionsPerGame() + (averageForcedFumblesPerGame()/5)) * 10;
      (c) getter 和 setter 方法
      (d) toString 方法：和 Quarterback 中类似，但要返回 Defensiveback 球员的信息

5. FootballTeam class:
   以数组形式存储 FootballPlayer 对象并实现基本的查询等操作
   (1) 实例变量：
      (a) team name
      (b) team owner
      (c) array of FootballPlayer objects（可以不填满）
      (d) 一个辅助变量标记当前球员数组添加到哪个下标了
   (2) 有参构造方法：
      设置 team name, team owner, 和 FootballPlayer 数组的大小
      创建 FootballPlayer 数组
      合理合适地初始化辅助变量的值
   (3) 其他方法：
      (a) addPlayer：将参数中的球员对象加入到球员数组中。请确保球员对象非空，球员数组还未满。如果球员对象为空、球员数组已满要打印出来做提示
      (b) findPlayerByPosition：返回字符串，将查询到某个位置的所有球员的信息打印出来。如果没有特定位置的球员，也要打印出没有查询到的提示。
      (c) toString 方法：打印球队信息，包括 team name, team owner, 所有球员的信息

6. FootballTester class:
   (1) 每个位置至少创建两个球员
   (2) 创建至少两个 FootballTeam 对象
   (3) 每个 FootballTeam 对象至少在每个位置上有一个球员
   (4) 用 compareTo 方法比较队员的排名
   (5) 测试所有类中的方法

核心代码展示:

FootballPlayer有参构造方法

```
    public FootballPlayer(String name,String position,String team,int
games_played){
        this.name=name;
        this.position=position;
        this.team=team;
        this.games_played=games_played;
    }
```

以Quaterback有参构造方法为例:

```
    public Quarterback(String name,String position,String team,int
games_played,int pass_attempts,int pass_completed,int touchdowns_passing,int
total_yards_passing){
        //子类的构造要首先构造父类
        super(name,position,team,games_played);
        this.pass_attempts=pass_attempts;
        this.pass_completed=pass_completed;
        this.touchdowns_passing=touchdowns_passing;
        this.total_yards_passing=total_yards_passing;
    }
```

其他函数按照要求构造即可，略

成果展示：

```
Information of 1st team

Team Name: team1, Team Owner: HarryPotter
Name: Phil Sims, Position: Quarter Back, Team: Giants, Completion Percentage:
0.7142857142857143, Average Passing Yards Per Game: 0.3, Average Touch Downs Per
Game: 0.8, Player's Rating: 71
Name: Jim Brown, Position: Running Back, Team: Browns, Average Yards Per Game:
0.5, Average Yards Per Attempt: 0.5833333333333334, Average Touch Downs Per
Game: 0.42857142857142855, Player's Rating: 1
Name: Spider Lockart, Position: Defensive Back, Team: Giants, Average Yards Per
Game: 1.0909090909090908, Average Yards Per Attempt: 0.2727272727272727, Average
Touch Downs Per Game: 0.18181818181818182, Player's Rating: 14

Information of 2nd team

Team Name: team2, Team Owner: TomRiddle
Name: Steve Young, Position: Quarter Back, Team: 49ers, Completion Percentage:
1.0, Average Passing Yards Per Game: 0.36363636363636365, Average Touch Downs
Per Game: 0.7272727272727273, Player's Rating: 100
Name: Saquon Barkley, Position: Running Back, Team: Giants, Average Yards Per
Game: 0.6363636363636364, Average Yards Per Attempt: 0.7777777777777778, Average
Touch Downs Per Game: 0.9090909090909091, Player's Rating: 1
Name: Aqib Talib, Position: Defensive Back, Team: Rams, Average Yards Per Game:
1.4285714285714286, Average Yards Per Attempt: 0.42857142857142855, Average
Touch Downs Per Game: 1.5714285714285714, Player's Rating: 21

Testing Finding player by position
Comparing Phil Simswith Aqib Talib
Phil Sims has a higher rating

Testing Finding player by position
Name: Jim Brown, Position: Running Back, Team: Browns, Average Yards Per Game:
0.5, Average Yards Per Attempt: 0.5833333333333334, Average Touch Downs Per
Game: 0.42857142857142855, Player's Rating: 1

Cannot find the FootballPlayer by the position
```

实验三

描述：井字棋是一种简单易懂的小游戏，我们将通过这个实验来学习用面向对象的设计思想来解决问题。我们知道，对于井字棋这个问题，面对过程的思路就是将其拆分为一个个步骤：

1.开始游戏

2.黑子先走

3.绘制画面

4.判断输赢

5.白子走...。

而对于面向对象的设计思想，我们就要以功能来划分问题，将整个五子棋分为

1.棋手类：他们继承自人，执黑/白子，能进行落子操作2.棋盘类：能接受落子操作，并绘制棋盘，判定输赢。

要求：

1.请用面向对象的思想设计上面提到的类：人，棋手，棋盘。要求考虑实际情况，各司其职，例如人有名字，棋
手继承自人，有黑白之分，可以落子。
2.设计一个测试类，利用上面设计的类及其功能，来开始一场对局，接受键盘的输入，进行对局。
3.图中展示了井字棋盘的样子，在实际设计中，你可以通过控制台的字符界面来输出一个棋盘，并接受用户的输
入来落子

实验思路：

Player类中落子方法makeDrop的参数需要包含Board，这样才可以实现在棋盘上落子的操作。Board类中检查有无赢家的check方法需要检查横着竖着和两条对角线，第一次编写时少些了竖着的情况，且检查对角线时不需要用循环，直接检查即可，且检查时要注意不能是横着竖着或两条对角线上都是"没有放置"的情况。

核心代码展示

```
#需包含Board
public int makeDrop(Board p, int x, int y) {
        return p.drop(this.isBlack, x, y);
}
```

```
#需检查横、竖、两条斜对角线
public int check() {
        for (int i = 0; i < 3; i++) {
            int temp = board[i][0];
            //System.out.println(temp);
            if (temp == 0) continue;
            boolean flag = true;
            for (int j = 1; j < 3; j++) {
                if (board[i][j] != temp) {
                    flag = false;
                    break;
                }
            }
            if (flag) {
                String winner = (temp == 1) ? "Black" : "White";
                System.out.println(winner + " wins!");
                return temp;
            }
        }
        for (int i = 0; i < 3; i++) {
            int temp = board[0][i];
            //System.out.println(temp);
```

```
            if (temp == 0) continue;
            boolean flag = true;
            for (int j = 1; j < 3; j++) {
                if (board[j][i] != temp) {
                    flag = false;
                    break;
                }
            }
            if (flag) {
                String winner = (temp == 1) ? "Black" : "White";
                System.out.println(winner + " wins!");
                return temp;
            }
        }
        int temp1 = board[0][0];
        if (temp1 != 0) {
            if (board[1][1] == temp1 && board[2][2] == temp1) {
                String winner = (temp1 == 1) ? "Black" : "White";
                System.out.println(winner + " wins!");
                return temp1;
            }
        }
        int temp2 = board[0][2];
        if (temp2 != 0) {
            if (board[1][1] == temp2 && board[2][0] == temp2) {
                String winner = (temp2 == 1) ? "Black" : "White";
                System.out.println(winner + " wins!");
                return temp2;
            }
        }
        System.out.println("Nobody wins now!");
        return 0;
}
```

实验四

实验四：从多边形到其他形状(正多边形，五角星)

描述：有很多其他的形状实际上可以用多边形来表达，例如正多边形， 五角星可以看作特殊的多边形，而圆则可以用多边形来拟合，在这个实验中，你会设计一个多边形类 Polygon，以及**继承自 Polygon 类**的正多边形类 RegularPolygon 和五角星类 Pentagram。

类的定义：

Point 类：包含公共的成员变量 x,y 和相应的公共构造方法和其余公共方法。

Polygon 类：

```
1. private Point[] points;  //类中唯一成员变量
2. public Polygon(Point[] points);  //类中唯一构造方法
3. public double getArea();     //返回面积
4. public double getPerimeter();    //返回周长
5. public int getSideNum(); //返回多边形的边数
```

RegularPolygon extends Polygon：

```
1. public RegularPolygon(double sideLength, int sideNum) //用边长和边数构造正多边形
2. public double getSideLength() //返回边长
```

Pentagram extends Polygon：

```
1. public Pentagram (double sideLength) //用边长构造五角星
```

要求：

1. 除以上的类定义之外，你可以**按实际需求和不同形状的实际属性**来定义更多的成员变量或重写父类的方法来更贴切的描述不同形状，并在 toString()方法中输出形状的信息。
2. 你可以任意假定多边形在坐标系中的位置。
3. 你可以假定 Polygon 类中多边形是凸多边形，但这样你就得在五角星类中 override getArea()方法，因为五角星是凹多边形。
4. 为了规范五角星的形状，我们规定五角星是一个正五边形把各角用直线相连并擦去原来的五边，并保留最外的线段。
5. 请编写**测试类** TestPolygon, TestRegularPolygon 和 TestPentagram 用于测试类的功能
6. 请在**测试类中**给出以下测试用例的结果（通过给定参数，构造对象，调用相应方法，给出面积，周长，边长，边数； 对于结果，请**保留 6 位小数**）

Polygon 测试用例：

1. (0, 0), (-1.5, 1.5), (1.5, 1.5)

---

2. (-0.5, 0), (0, 1),(2, 1), (2.5, 0)
3. (-4.22, -0.82), (-3.26, 2.24), (-0.6, 4.02), (2.8, 2.6), (1.1, -1.68)

RegularPolygon 测试用例：

1. sideLength:3.1415926, sideNum:3
2. sideLength:3.1415926, sideNum:12
3. sideLength:3.1415926, sideNum:1000

Pentagram 测试用例：

1. sideLength:1.0

实验思路：

查阅资料可知，若为凸多边形，可直接由坐标解得面积和周长，公式如下

1. 普通多边形

- 面积公式

$$S=\sum_{k=1}^{\infty} S_{\triangle op_kp_{k+1}}=\frac{1}{2}\sum_{k=1}^{\infty}(x_ky_{k+1}-x_{k+1}y_k)$$

- 周长公式

计算每条边的欧拉距离即可

2. 正多边形

- 面积公式

$$A = \frac{nt^2 sin(\frac{2\pi}{n})}{4[1 - cos(\frac{2\pi}{n})]}$$

- 周长

算出一条边的边长*n

3. 五角星

- 面积公式

$$S = \frac{5l^2}{ctg36 + ctg18}$$

- 周长公式

$$10l$$

核心代码展示

```
#普通多边形
public double getArea() {
        double res = 0;
        for (int i = 0; i < this.points.length - 1; i++) {
            res += (this.points[i].x * this.points[i + 1].y - this.points[i +
1].x * this.points[i].y) / 2.0;
        }
        res += (this.points[this.points.length - 1].x * this.points[0].y -
this.points[0].x * this.points[this.points.length - 1].y) / 2.0;
        return Math.abs(res);
    }
    public double getPerimeter() {
        double res = 0;
        for (int i = 0; i < this.points.length - 1; i++) {
            res += Math.abs(Math.sqrt((this.points[i].x - this.points[i + 1].x)
* (this.points[i].x - this.points[i + 1].x) + (this.points[i].y - this.points[i
+ 1].y) * (this.points[i].y - this.points[i + 1].y)));
        }
        res += Math.abs(Math.sqrt((this.points[this.points.length - 1].x -
this.points[0].x) * (this.points[this.points.length - 1].x - this.points[0].x) +
(this.points[this.points.length - 1].y - this.points[0].y) *
(this.points[this.points.length - 1].y - this.points[0].y)));
        return res;
    }
```

```
#正多边形
public double getArea() {
        double res = this.sideNum * 1.0 * this.sideLength * this.sideLength *
Math.sin(2 * Math.PI / this.sideNum) /
                (4 * (1 - Math.cos(2 * Math.PI / this.sideNum)));
        return res;
    }
    public double getPerimeter() {
        double res = this.sideNum * this.sideLength * 1.0;
        return res;
}
```

```
# 五角星
public double getArea() {
        double res = 5 * this.sideLength * this.sideLength / (1.0 /
Math.tan(Math.PI / 5.0) + 1.0 / Math.tan(Math.PI / 10.0));
        return res;
    }
    public double getPerimeter() {
        double res = 10 * this.sideLength;
        return res;
    }
（4）用df.setMinimumFractionDigits设置小数位数
public String toString() {
        DecimalFormat df = new DecimalFormat();
        df.setMinimumFractionDigits(6);
        return "Area: " + df.format(this.getArea()) +
                " Perimeter: " + df.format(this.getPerimeter()) +
                " SideNum: " + this.getSideNum();
    }
```