day06——Java编程案例（专题）

各位同学，前面我们已经学习过很多Java的基础知识了，主要有**变量、数组、运算符、流程控制、方法等。**但是对于这些知识点的运用，掌握得还不是很熟练，所以今天我们专门花一天时间，给同学们讲几个专项练习题，把前面所学习的知识巩固一下。

同时通过这些专项练习题，**积攒大家的代码量，以便提升大家的编程能力和编程思维。**这里所说的编程思维就是使用Java技术解决问题的思维方式；编程能力就是按照编程思维编写代码的能力。

想要提升编程思维和编程能力，在这里给同学们一些学习上的建议：

- 编程思维、编程能力不是一朝一夕形成的，需要大量思考，练习和时间的沉淀。
- 具体措施：前期，建议先模仿；后期，自然就能创新了；
  勤于练习代码，勤于思考，孰能生巧。

中国的航空母舰、战斗机，这些技术都是先模仿，再创新的，而且的模仿的周期是非常长的。所以同学们在使用Java技术解决问题时，也是先模仿一些特定问题的解决思路，以后遇到同类型的问题，就采用同一种思维模式来做就行。

案例一：买飞机票

各位同学，我们先来学习第一个案例《飞机买票》，先仔细阅读一下案例需求

案例 买飞机票

需求

用户购买机票时，机票原价会按照淡季、旺季，头等舱还是经济舱的情况进行相应的优惠，优惠方案如下：5-10月为旺季，头等舱9折，经济舱8.5折； 11月到来年4月为淡季，头等舱7折，经济舱6.5折，请开发程序计算出用户当前机票的优惠价。

我们来分析一下，这个需求该如何实现。前面我跟同学们讲过，将来我们去做一些需求，都是一个一个方法来实现的，所以在这里我们也采用方法来编写。

这个方法如何编写呢？采用下面的方式来思考

1.首先，考虑方法是否需要接收数据处理？
    阅读需求我们会发现，不同月份、不同原价、不同舱位类型优惠方案都不一样；
    所以，可以将原价、月份、舱位类型写成参数

2.接着，考虑方法是否有返回值？
    阅读需求我们发现，最终结果是求当前用户的优惠票价
    所以，可以将优惠票价作为方法的返回值。

3.最后，再考虑方法内部的业务逻辑
    先使用`if`判断月份是旺季还是淡季，然后使用`switch`分支判断是头等舱还是经济舱，计算　　票价

代码如下

```
public class Test1 {
    public static void main(String[] args) {
        // 目标：完成买飞机票的案例。
        double price = calculate(1000, 11, "头等舱");
        System.out.println("优惠价是：" + price);
    }

    public static double calculate(double price,int month,String type){
        // 1、判断当前月份是淡季还是旺季
        if(month >= 5 && month <= 10) {
            // 旺季
            // 2、判断仓位类型。
            switch (type){
                case "头等舱":
                    price *= 0.9; // price = price * 0.9;
                    break;
                case "经济舱":
                    price *= 0.85;
                    break;
            }
        }else {
            // 淡季
            switch (type){
                case "头等舱":
                    price *= 0.7; // price = price * 0.7;
                    break;
                case "经济舱":
                    price *= 0.65;
```

```
                    break;
                }
            }
            return price;
        }
    }
```

案例二：开发验证码

各位同学，接下来，我们学习第二个案例《开发验证码》，同样先阅读一下案例需求

分析一下，需求是要我们开发一个程序，生成指定位数的验证码。考虑到实际工作中生成验证码的功能很多地方都会用到，为了提高代码的复用性，我们还是把生成验证码的功能写成方法比较好。

那生成验证码的方法该怎么写呢？按照下面的三个步骤进行思考

```
1.首先，考虑方法是否需要接收数据处理？
    要求生成指定位数的验证码，到底多少位呢？让调用者传递即可
    所以，需要一个参数，用来表示验证码的位数

2.接着，考虑方法是否需要有返回值？
    该方法的结果，就是为了得到验证码
    所以，返回值就是验证码；

3.最后，再考虑方法内部的业务逻辑
    1)先按照方法接收的验证码位数n,循环n次
    2)每次循环，产生一个字符，可以是数字字符、或者大小写字母字符
    3)定义一个String类型的变量用于记住产生的每位随机字符
```

按照思路，编写代码如下

```
public class Test2 {
    public static void main(String[] args) {
        // 目标：完成生成随机验证码。
        System.out.println(createCode(8));
    }

    public static String createCode(int n){
        //1)先按照方法接收的验证码位数n,循环n次
        Random r = new Random();
        //3)定义一个String类型的变量用于记住产生的每位随机字符
        String code = "";
```

```
        for (int i = 1; i <= n; i++) {
            // i = 1 2 3 4 5
            //2)每次循环，产生一个字符，可以是数字字符、或者大小写字母字符
            // 思路：随机一个0 1 2之间的数字出来，0代表随机一个数字字符，1、2代表随机大写字母，小写字母。
            int type = r.nextInt(3); // 0 1 2
            switch (type) {
                case 0:
                    // 随机一个数字字符
                    code += r.nextInt(10); // 0 - 9  code = code + 8
                    break;
                case 1:
                    // 随机一个大写字符 A 65   Z 65+25    (0 - 25) + 65
                    char ch1 = (char) (r.nextInt(26) + 65);
                    code += ch1;
                    break;
                case 2:
                    // 随机一个小写字符 a 97   z 97+25    (0 - 25) + 97
                    char ch2 = (char) (r.nextInt(26) + 97);
                    code += ch2;
                    break;
            }
        }
        return code;
    }
}
```

案例三：评委打分

各位同学，接下来，我们学习第三个案例《评委打分》，同样先阅读一下案例需求

案例 评委打分案例

需求

在唱歌比赛中，可能有多名评委要给选手打分，分数是[0 - 100]之间的整数。选手最后得分为：去掉最高分、最低分后剩余分数的平均分，请编写程序能够录入多名评委的分数，并算出选手的最终得分。

我们把上面的需求还是用方法来编写。

1.首先，考虑方法是否需要接收数据来处理？
    需求中说，有多个评委的打分，但是到底多少个评委呢？ 可以由调用者传递
    所以，我们可以把评委的个数写成参数；

2.接着，考虑方法是否需要有返回值？
    需求中，想要的最终结果是平均分
    所以，返回值就是平均分；

3.最后，再考虑方法内部的业务逻辑
    1)假设评委的个位为n个，那么就需要n个评委的分数，首先可以新建一个长度为n的数组，用来存储每一个评委的分数

    2)循环n次，使用Scanner键盘录入n个1~100范围内的整数，并把整数存储到数组中

3)求数组中元素的总和、最大值、最小值

4)最后再计算平均值； 平均值 = (和-最大值-最小值)/(数组.length-2);

代码如下

```
public class Test3 {
    public static void main(String[] args) {
        // 目标：完成评委打分案例。
        System.out.println("当前选手得分是：" + getAverageScore(6));
    }

    public static double getAverageScore(int n){
        // 1、定义一个动态初始化的数组，负责后期存入评委的打分
        int[] scores = new int[n]; // 6
        // scores = [0, 0, 0, 0, 0, 0]

        // 2、遍历数组的每个位置，依次录入评委的分数
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < scores.length; i++) {
            // i = 0 1 2 3 4 5
            System.out.println("请您录入第"+ (i + 1) +"个评委的分数：");
            int score = sc.nextInt();
            scores[i] = score;
        }

        // 3、从数组中计算出总分，找出最高分，最低分。
        int sum = 0; // 求总分用的变量
        int max = scores[0]; // 求最大值的
        int min = scores[0]; // 求最小值的。

        // 遍历数组找出这些数据的。
        for (int i = 0; i < scores.length; i++) {
            // i = 0 1 2 3 4 5
            int score = scores[i];
            // 求和
            sum += score;
            // 求最大值
            if(score > max){
                max = score;
            }
            // 求最小值
            if(score < min){
                min = score;
            }
        }
        // 4、计算出平均分并返回
        return 1.0 * (sum - min - max) / (number - 2);
    }
}
```

案例四：数字加密

各位同学，接下来我们学习第四个案例《数字加密》，我们还是先阅读一下案例需求

案例 数字加密

需求

某系统的数字密码是一个四位数，如1983，为了安全，需要加密后再传输，加密规则是：对密码中的每位数，都加5 ,再对10求余，最后将所有数字顺序反转，得到一串加密后的新数，请设计出满足本需求的加密程序！

仔细阅读需求后发现，简答来说该需求要做的事情，就是把一个4位数的整数，经过一系列的加密运算（至于怎么运算，待会再详细分析），得到一个新的整数。

我们还是把这个需求用方法来实现，按照下面的思维模式进行分析

1.首先，考虑方法是否需要接收数据处理？
    需要一个4位数，至于是哪一个数，让方法的调用者传递。
    所以，方法的参数，就是这个需要加密的四位数

2.接着，考虑方法是否需要有返回值？
    方法最终的结果是一个加密后的数据
    所以，返回值就表示为加密后的数据。

3.最后，再考虑方法内部的业务逻辑，这里的业务逻辑就是那一系列的加密运算
    1)先要把4位数整数拆分为，4个数字，用一个数组保存起来
    2)再将数组中的每一个元素加5，再对10取余
    3)最后将数组中的元素反转，

```
public class Test4 {
    public static void main(String[] args) {
        // 目标：完成数字加密程序的开发。
        System.out.println("加密后的结果是：" + encrypt(8346));
    }

    public static String encrypt(int number){
        // number = 1983
        // 1、把这个密码拆分成一个一个的数字，才可以对其进行加密啊。
        int[] numbers = split(number);
        // numbers = [1, 9, 8, 3]

        // 2、遍历这个数组中的每个数字，对其进行加密处理。
        for (int i = 0; i < numbers.length; i++) {
            // i = 0 1 2 3
            numbers[i] = (numbers[i] + 5) % 10;
        }
        // numbers = [6, 4, 3, 8]

        // 3、对数组反转，把对数组进行反转的操作交给一个独立的方法来完成
        reverse(numbers);
        // numbers = [8, 3, 4, 6]

        // 4、把这些加密的数字拼接起来做为加密后的结果返回即可。
        String data = "";
        for (int i = 0; i < numbers.length; i++) {
            data += numbers[i];
        }
```

```
        return data;
    }

    public static void reverse(int[] numbers) {
        // 反转数组的
        // numbers = [6, 4, 3, 8]
        //            i        j
        for (int i = 0, j = numbers.length - 1; i < j; i++,j--) {
            // 交换i和j位置处的值。
            // 1、把后一个位置处的值交给一个临时变量先存起来
            int temp = numbers[j];
            // 2、把前一个位置处的值赋值给后一个位置处
            numbers[j] = numbers[i];
            // 3、把后一个位置处原来的值（由临时变量记住着）赋值给前一个位置
            numbers[i] = temp;
        }
    }

    public static int[] split(int number) {
        // number = 1983
        int[] numbers = new int[4];
        numbers[0] = number / 1000;
        numbers[1] = (number / 100) % 10;
        numbers[2] = (number / 10) % 10;
        numbers[3] = number % 10;
        return numbers;
    }
}
```

案例五：数组拷贝

各位同学，接下来我们学习第五个案例《数组拷贝》，我们还是先阅读一下案例需求

仔细阅读需求发现，想要实现的效果就是：给定一个数组，然后经过我们编写的程序，得到一个和原数组一模一样的数组。

我们也采用一个方法来编写，按照下面的思维模式来思考

1.首先，考虑方法是否需要接收数据处理？
    该方法的目的是拷贝数组，拷贝哪一个数组呢？ 需要调用者传递
    所以，参数应该是一个数组

2.接着，考虑方法是否需要有返回值？
    该方法最终想要得到一个新数组
    所以，返回值是拷贝得到的新数组

3.最后，考虑方法内部的业务逻辑？
    1)创建一个新的数组，新数组的长度和元素数组一样
    2)遍历原数组，将原数组中的元素赋值给新数组
    3)最终将新数组返回

```
public class Test5 {
    public static void main(String[] args) {
        // 目标：掌握数组拷贝。
        int[] arr = {11, 22, 33};
        int[] arr2 = copy(arr);
        printArray(arr2);

        // 注意：这个不是拷贝数组，叫把数组变量赋值给另一个数组变量。
        //         int[] arr3 = arr;
        //         arr3[1] = 666;
        //         System.out.println(arr[1]);

        arr2[1] = 666;
        System.out.println(arr[1]);
    }

    public static int[] copy(int[] arr){
        // arr = [11, 22, 33]
        //         0    1   2

        // 1、创建一个长度一样的整型数组出来。
        int[] arr2 = new int[arr.length];
        // arr2 = [0, 0, 0]
        //          0  1  2

        // 2、把原数组的元素值对应位置赋值给新数组。
        for (int i = 0; i < arr.length; i++) {
            // i = 0 1 2
            arr2[i] = arr[i];
        }

        return arr2;
    }

    public static void printArray(int[] arr){
        System.out.print("[");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(i==arr.length-1 ? arr[i] : arr[i] + ", ");
        }
        System.out.println("]");
    }
}
```

案例六：抢红包

各位同学，接下来我们学习第六个案例《抢红包》，我们还是先阅读一下案例需求

案例 抢红包

需求

一个大V直播时发起了抢红包活动，分别有：9、666、188、520、99999五个红包。 请模拟粉丝来抽奖，按照先来先得，随机抽取，抽完即止，注意：一个红包只能被抽一次，先抽或后抽哪一个红包是随机的，示例如下（不一定是下面的顺序）：

我们还是把这个案例用一个方法来编写，同样按照下面的模式来分析

1.首先，考虑方法是否需要接收数据处理？
需要接收5个红包，至于是哪5个红包，可以有调用者传递；把5个红包的数值，用数组来存储。
所以，参数就是一个数组

2.接着，考虑方法是否需要有返回值？
按照需求的效果，抢完红包就直接打印了，不需要返回值

3.最后，考虑方法内部的业务逻辑是怎么的？

思考：红包实际上是数组中的元素，抢红包实际上随机获取数组中的元素；而且一个红包只能抢一次，怎么做呢？我们可以把数组中获取到元素的位置，置为0,下次再或者这个位置的元素一判断为0，再重新获取新的元素，依次内推，直到把数组中所有的元素都获取完。

我们我们把抽红包的思路再整理一下：

1)首先，写一个循环，循环次数为数组的长度

2)每次循环，键盘录入，提示"用户录入任意键抽奖："

3)随机从数组中产生一个索引，获取索引位置的元素，这个元素就表示抽的红包

如果值不为0，则打印如："恭喜您，您抽中了520元",把这个位置元素置为0

如果值为0，则说明这个红包被抽过，重新循环到第2步，重新抽奖

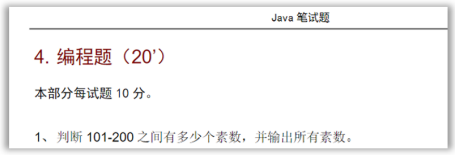
【注意：如果当前这一次没有抽中，这一次抽奖机会被浪费掉了，我们可以把控制循环的次数自减一下】

```
public class Test6 {
    public static void main(String[] args) {
        int[] moneys = {100,999,50,520,1314};
        start(moneys);
    }
    //开始抽奖
    public static void start(int[] moneys){
        //1)首先，写一个循环，循环次数为数组的长度
        for (int i = 0; i < moneys.length; i++) {
            //2)每次循环，键盘录入，提示"用户录入任意键抽奖："
            while (true){
                Scanner sc = new Scanner(System.in);
                System.out.print("用户录入任意键抽奖：");
                String msg = sc.next();
                //3)随机从数组中产生一个索引，获取索引位置的元素，这个元素就表示抽的红包
                Random r = new Random();
                int index = r.nextInt(moneys.length);
                int money = moneys[index];
                if(money!=0){
                    //如果值不为0，则打印如："恭喜您，您抽中了520元"
                    System.out.println("恭喜您，您抽中了"+money+"元");
                    moneys[index] = 0;
                    break;
                }else {
                    //如果值为0，则说明这个红包被抽过，重新循环到第2步，重新抽奖
                    //此时这一次抽奖机会被浪费掉了，可以把控制循环的次数自减一下
                    i--;
                }
            }
        }
    }
}
```

案例七：找素数

各位同学，接下来我们学习第七个案例《找素数》，我们还是先阅读一下案例需求

案例 找素数


Java 笔试题

4. 编程题（20’）

本部分每试题 10 分。

1、判断 101-200 之间有多少个素数，并输出所有素数。

说明：除了1和它本身以外，不能被其他正整数整除，就叫**素数。**

比如：3、7就是素数，而9、21等等不是素数。

首先我们得统一认识一下什么是素数：**只能被1和本身整除的数是素数**，比如：3、7是素数，9,21不是素数（因为9可以被3整除，21可以被3和7整除）

再思考题目需求该怎么做？**打印输出101~200之间的素数，并求有多少个？**，我们也是把这个需求写成一个方法，还是按照三个步骤分析方法如何编写。

1.首先，考虑方法是否需要接收数据处理？
　　该方法是求一个范围内的素数，一个范围需要两个数据来确定，比如：101~200
　　所以，方法需要两个参数来接收范围的开始值start，和范围的结束值end

2.接着，考虑方法是否需要返回值？
　　该方法需要求一个范围内的素数的个数
　　所以，返回值就是素数的个数

3.最后，考虑方法内部的业务逻辑
　　思考：怎么判断一个数是素数呢？要仅仅抓住，素数的要求:“只能被1和本身整除的数是素数”。我们可以从反向思考，如果这个数只要能被除了1和本身以外的数整除，那么这个数就不是素数。

```
//比如1：判断9是否为素数
int num = 9;
boolean flag = true; //规定flag等于true表示num是素数；否则表示num不是素数
//如果这个数num只要能被除了1和本身以外的数整除，那么这个数就不是素数。
for(int j=2; j<9-1; j++){
    //当j=3时，num%j == 9%3 == 0;
    if(num%j==0){
        //说明num=9; 表示一个素数。把flag改为false;
        flag = false;
    }
}
```

　　把上面的代码循环执行，每次循环然后把num换成start~end之间的整数即可。

编写代码如下

```
public class Test7 {
    public static void main(String[] args) {
        // 目标：完成找素数。
        System.out.println("当前素数的个数是：" + search(101, 200));
    }

    public static int search(int start, int end){
        int count = 0;
        // start = 101   end = 200
        // 1、定义一个for循环找到101到200之间的每个数据
        for (int i = start; i <= end ; i++) {
            // i = 101 102 103 ... 199 200

            // 信号位思想
            boolean flag = true; // 假设的意思：默认当前i记住的数据是素数。
```

```
            // 2、判断当前i记住的这个数据是否是素数。
            for (int j = 2; j <= i / 2; j++) {
                if(i % j == 0){
                    // i当前记住的这个数据不是素数了
                    flag = false;
                    break;
                }
            }
            // 3、根据判定的结果决定是否输出i当前记住的数据：是素数才输出展示。
            if(flag){
                System.out.println(i);
                count++;
            }
        }
        return count;
    }
}
```

案例八：模拟双色球[拓展案例]

各位同学，接下来我们学习第八个案例《模拟双色球》，我们还是先阅读一下案例需求

案例 模拟双色球：业务分析、用户投注一组号码

双色球业务介绍：

需求

双色球彩票，由前区6个红球号码和后区1个蓝球号码组成。前区6个红球号码是从1~33中选择；后区蓝球号码是从1~16中选择；而且前区6个红球号码是不能重复的。 编写程序，能够让用户录入一注双色球彩票；也可以随机产生一注双色球彩票

这个案例我们可以采用方法方法来完成

1. 第一个方法，让用户手动投注，产生一注双色球彩票

```
/** 1、用于让用户投注一组号码（前6个是红球，最后1个是蓝球），并返回用户投注的号码 */
public static int[] userSelectNumbers(){...}
```

2. 第二个方法，由系统随机产生一注双色球彩票开奖号码

```
/** 2、系统随机一组中奖号码（前6个是红球，最后1个是蓝球），并返回这组中奖号码 */
public static int[] createLuckNumbers(){...}
```

3. 第三个方法，判断传入两组号码，用于判断彩票的中奖情况

```
/** 3、传入两组号码，用来判断用户投注号码的中奖情况，并输出 */
public static void judge(int[] userNumbers, int[] luckNumbers){...}
```

8.1 手动投注

编写一个方法，让用户手动投注，产生一注双色球彩票，思路分析

```
1.首先，考虑方法是否需要接收数据处理？
    双色球彩票的规则非常明确，没有什么数据需要传递给方法。
    所以，不需要参数

2.接着，考虑方法是否需要返回值？
    方法最终的结果是需要一注双色球彩票的号码，一注彩票有7个号码，可以用一个数组来存
    所以，返回值是一个数组

3.最后，考虑方法内部的业务逻辑怎么编写？
    1)首先需要准备一个int类型数组，长度为7；用于存储产生的投注号码
    2)循环遍历数组的前6个元素，采用键盘录入的方式，给前区6个红球赋值
        要求录入的整数在1~33范围内，同时录入的整数在数组中不能已存在，否则重新录入
    3)最后再录入一个整数，给后区一个蓝球赋值
        要求整数必须在1~16范围内
```

- 手动投注代码如下

```
/** 1、设计一个方法，用于让用户投注一组号码并返回（前6个是红球号码，最后1个是蓝球号码 ）*/
public static int[] userSelectNumbers(){
    // 2、创建一个整型数组，用于存储用户投注的7个号码（前6个是红球号码，最后1个是蓝球号码 ）
    int[] numbers =  new int[7];
    // numbers = [0, 0, 0, 0, 0, 0, 0]
    //            0  1  2  3  4  5  6

    Scanner sc = new Scanner(System.in);
    // 3、遍历前6个位置，让用户依次投注6个红球号码，存入
    for (int i = 0; i < numbers.length - 1; i++) {
        // i = 0 1 2 3 4 5

        while (true) {
            // 4、开始让用户为当前位置投注一个红球号码（1-33之间，不能重复）
            System.out.println("请您输入第" + (i + 1) + "个红球号码（1-33之间，不能重
复）：");
            int number = sc.nextInt();

            // 5、先判断用户输入的红球号码是否在1-33之间
            if(number < 1 || number > 33){
                System.out.println("对不起，您输入的红球号码不在1-33之间，请确认！");
            }else {
                // 号码是在1-33之间了，接着还要继续判断这个号码是否重复，不重复才可以使用。
                if(exist(numbers, number)){
                    // number当前这个红球号码是重复了。
                    System.out.println("对不起，您当前输入的红球号码前面选择过，重复了，
请确认！");
                }else {
                    // number记住的这个号码没有重复了，就可以使用了。
                    numbers[i] = number;
                    break; // 结束当次投注，结束了当前死循环。
                }
            }
        }
    }
```

```
    // 6、投注最后一个蓝球号码。
    while (true) {
        System.out.println("请您输入最后1个蓝球号码（1-16）：");
        int number = sc.nextInt();
        if(number < 1 || number > 16){
            System.out.println("对不起，您输入的蓝球号码范围不对！");
        }else {
            numbers[6] = number;
            break; // 蓝球号码录入成功，结束死循环
        }
    }

    return numbers;
}
```

每键盘录入一个号码，需要判断这个号码在数组中是否存在，存在返回true；不存在返回false

```
private static boolean exist(int[] numbers, int number) {
    // 需求：判断number这个数字是否在numbers数组中存在。
    // numbers = [12, 25, 18, 0, 0, 0, 0]
    // number = 12
    for (int i = 0; i < numbers.length; i++) {
        if(numbers[i] == 0){
            break;
        }
        if(numbers[i] == number){
            return true;
        }
    }
    return false;
}
```

为了打印一注彩票的号码（数组中的元素），把打印数组中的元素也写成方法。

```
public static void printArray(int[] arr) {
    System.out.print("[");
    for (int i = 0; i < arr.length; i++) {
        System.out.print(i == arr.length - 1 ? arr[i] : arr[i] + ", ");
    }
    System.out.println("]");
}
```

在main方法中测试，运行看能不能产生一注彩票号码

```
public class Test8 {
    public static void main(String[] args) {
        // 目标：完成双色球系统的开发。
        int[] userNumbers = userSelectNumbers();
        System.out.println("您投注的号码：");
        printArray(userNumbers);
    }
}
```

8.2 随机开奖号码

编写一个方法，让用户自动机选投注，产生一注双色球彩票，思路分析

```
1.首先，考虑方法是否需要接收数据处理？
    双色球彩票的规则非常明确，没有什么数据需要传递给方法。
    所以，不需要参数

2.接着，考虑方法是否需要返回值？
    方法最终的结果是需要一注双色球彩票的号码，一注彩票有7个号码，可以用一个数组来存
    所以，返回值是一个数组

3.最后，考虑方法内部的业务逻辑怎么编写？
    1)首先需要准备一个int类型数组，长度为7；用于存储产生的投注号码
    2)循环遍历数组的前6个元素，采用生成随机数的的方式，给前区6个红球赋值
        要求生成的随机数在1~33范围内，同时随机的整数数组中不能已存在，否则重新生产
    3)最后再随机一个整数，给后区一个蓝球赋值
        要求随机整数必须在1~16范围内
```

机选号码，代码如下

```
/** 2、设计一个方法：随机一组中奖号码出来（6个红球号码，1个蓝球号码 ）*/
public static int[] createLuckNumbers(){
    // 1、创建一个整型数组，用于存储这7个号码
    int[] numbers = new int[7];

    Random r  = new Random();
    // 2、遍历前6个位置处，依次随机一个红球号码存入（1-33 不重复）
    for (int i = 0; i < numbers.length - 1; i++) {
        // i = 0 1 2 3 4 5
        while (true) {
            // 3、为当前这个位置随机一个红球号码出来存入。
            //1 - 33 ==> -1 ===> (0 , 32) + 1
            int number = r.nextInt(33) + 1;

            // 4、判断这个号码是否之前出现过（红球号码不能重复）。
            if(!exist(numbers, number)){
                // number不重复。
                numbers[i] = number;
                //结束死循环，代表找到了当前这个位置的一个不重复的红球号码了。
                break;
            }
        }
    }

    // 3、录入一个蓝球号码。 1-16
    numbers[6] = r.nextInt(16) + 1;
    return numbers;
}
```

在main方法中测试，看是否能够产生一注彩票

```
public class Test8 {
    public static void main(String[] args) {
        // 目标：完成双色球系统的开发。
        //用户手动投注
```

```
        int[] userNumbers = userSelectNumbers();
        System.out.println("您投注的号码：");
        printArray(userNumbers);

        //生成中奖号码
        int[] luckNumbers = createLuckNumbers();
        System.out.println("中奖的号码：");
        printArray(luckNumbers);
    }
}
```

8.3 判断是否中奖

编写一个方法，判断用户的彩票号码是否中奖，具体中奖规则如下

- 6个红球+1个蓝球 ，奖金1000万
- 6个红球+0个蓝球，奖金500万
- 5个红球+1个蓝球，奖金3000块
- 5个红球+0个蓝球，或者4个红球+1个蓝球，奖金200块
- 4个红球+0个蓝球，或者3个红球+1个蓝球，奖金10块
- 小于3个红球+1个蓝球，奖金5块
- 如果前面的都不成立，就中奖，算你为福利事业做贡献了。

编写方法的思路如下

```
1.首先，考虑方法是否需要接收数据处理？
    判断彩票是否中奖，需要有两组号码；一组号码是彩票号码，一组号码是开奖号码
    所以，参数需要有两个数组

2.接着，考虑方法是否需要返回值？
    方法不需要返回结果，中了奖，直接将奖项打印输出就行了。
    【注意：这只是提供一种代码的编写方案，你将中奖的金额返回也行】

3.最后，考虑方法内部的业务逻辑怎么编写？
    1)定义两个变量redCount和blueCount用来记录，红球的个数和蓝球的个数
    2)遍历两个数组中前6个元素(红球)，判断两个数组中有没有相同元素
        如果找到一个相同元素，则redCount++
    3)比较两个数组中最后一个元素(蓝球)是否相同
        如果相同，则blueCount++
    4)根据红球和蓝球的命中个数，打印输出对应的奖项
```

代码如下

```
/** 3、设计一个方法，用于判断用户的中奖情况 */
public static void judge(int[] userNumbers,int[] luckNumbers){
    // userNumbers = [12, 14, 16, 18, 23, 26, 8]
    // luckNumbers = [16, 17, 18, 19, 26, 32, 8]

    // 2、分别定义2个变量用于记住红球命中了几个以及蓝球命中了几个
    int redCount = 0;
    int blueCount = 0;

    // 先判断红球命中的数量。
    // 遍历用户投注的号码的前6个红球
```

```
    for (int i = 0; i < userNumbers.length - 1; i++) {
        // userNumbers[i]
        // 开始遍历中奖号码的前6个红球号码，看用户当前选择的这个号码是否命中了
        for (int j = 0; j < luckNumbers.length - 1; j++) {
            if(userNumbers[i] == luckNumbers[j]){
                redCount++;
                break;
            }
        }
    }

    // 3、判断蓝球是否命中了
    blueCount = userNumbers[6] == luckNumbers[6] ? 1 : 0;

    System.out.println("您命中的红球数量是：" + redCount);
    System.out.println("您命中的蓝球数量是：" + blueCount);

    // 4、判断中奖详情，并输出结果
    if(redCount == 6 && blueCount == 1){
        System.out.println("恭喜您，中奖1000万，可以开始享受人生了~~~");
    }else if(redCount == 6 && blueCount == 0){
        System.out.println("恭喜您，中奖500万，可以稍微开始享受人生了~~~");
    }else if(redCount == 5 && blueCount == 1){
        System.out.println("恭喜您，中奖3000元，可以出去吃顿小龙虾了~");
    }else if(redCount == 5 && blueCount == 0 || redCount == 4 && blueCount == 1)
{
        System.out.println("恭喜您，中了小奖：200元~");
    }else if(redCount == 4 && blueCount == 0 || redCount == 3 && blueCount == 1)
{
        System.out.println("中了10元~");
    }else if( redCount < 3 && blueCount == 1){
        System.out.println("中了5元~");
    }else {
        System.out.println("感谢您对福利事业做出的巨大贡献~~~");
    }
}
```

在main方法中测试，检测是否中奖的方法是否正确

```
public class Test8 {
    public static void main(String[] args) {
        // 目标：完成双色球系统的开发。
        //用户投注
        int[] userNumbers = userSelectNumbers();
        System.out.println("您投注的号码：");
        printArray(userNumbers);

        //随机产生一个中奖号码
        int[] luckNumbers = createLuckNumbers();
        System.out.println("中奖的号码：");
        printArray(luckNumbers);

        //判断用户投注的号码是否中奖
        judge(userNumbers, luckNumbers);
    }
}
```