Java数组基础作业题

选择题

1. (单选题) 下列关于数组动态初始化的描述中，错误的是：C
   A、动态初始化时系统会为数组元素分配默认值
   B、动态初始化格式为：数据类型[] 数组名 = new 数据类型[数组长度]
   C、动态初始化时可以直接指定数组元素的值
   D、int[] arr = new int[5]; 创建了一个可以存储5个整数的数组
2. (单选题) 以下代码的输出结果是：C

```
int[] arr = {10, 20, 30, 40, 50};
System.out.println(arr[2]);
```

   A、10
   B、20
   C、30
   D、40
3. (多选题) 关于数组索引的描述，正确的有：ABD
   A、数组索引从0开始
   B、可以通过数组名[索引]的方式访问数组元素
   C、数组的最大索引是数组的长度
   D、访问不存在的索引会抛出ArrayIndexOutOfBoundsException异常
4. (多选题) 以下哪种方式可以正确初始化一个包含3个元素的字符串数组：BD
   A、String[] arr = new String[3]{};
   B、String[] arr = new String[3];
   C、String[] arr = new String(3);
   D、String[] arr = {"a", "b", "c"};
5. (多选题) 关于数组的内存分配，下列说法正确的是：ABCD
   A、数组是存储在堆内存中的
   B、数组变量存储在栈内存中
   C、多个数组变量可以指向同一个数组对象
   D、数组一旦创建，长度就不能改变

问答题

1. 请简要说明数组静态初始化和动态初始化的区别，并各举一个例子。

1、静态初始化，可以直接设置数组中的元素值，动态初始化不可以。2、动态初始化可以只设置数组长度，静态初始化在无元素的情况下数组长度只能为0

1. 编写一个数组遍历的代码时，为什么通常使用循环而不是逐个打印每个元素？请说明原因。

遍历数组的目的是找出需要的数据，打印每个元素毫无意义。循环代码简单 速度快

编程题

1. 编写一个Java程序，实现以下功能：

- 创建一个长度为10的整型数组
- 使用Random类生成10个0-100之间的随机整数，存入数组
- 找出数组中的最大值和最小值
- 计算并输出所有元素的平均值
- 统计数组中大于平均值的元素个数

要求：

1. 使用数组的动态初始化
2. 使用循环结构完成数组的遍历和操作
3. 输出格式如下：

```
数组元素：[...]
最大值：xx
最小值：xx
平均值：xx
大于平均值的元素个数：xx
```

```java
package zuoye;

import java.util.Random;

public class exam {
    public static void main(String[] args) {
        int[] arr = new int[10];
        Random random = new Random();
        int max=0,minn=9999999,sum=0;
        double avg = 0;

        for (int i = 0; i < arr.length; i++) {
            arr[i]= random.nextInt(0,9)+1;
            if(arr[i]>max){
                max=arr[i];
            }else if(arr[i]<minn){
                minn = arr[i];
            }
            sum+=arr[i];
        }

        avg = sum/10;
        int count=0;
        for (int i = 0; i < arr.length; i++) {
            if(arr[i]>avg){
                count++;
            }
        }
        System.out.println("数组元素为：");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(" "+arr[i]);
        }

        System.out.println();
        System.out.println("最大值:"+max);

        System.out.println("最大值:"+minn);
```

```
        System.out.println("平均值:"+avg);

        System.out.println("大于平均值的元素个数:"+count);


    }
}
```