**04-数组-作业**

1. 使用多种不同方式创建数组并初始化，数组包含a、b、c三个元素

**答：**

1.     public static void main(String[] args) {

2.         //1、先声明再赋值

3.         char[] arr1 = new char[3];

4.         arr1[0] = 'a';

5.         arr1[1] = 'b';

6.         arr1[2] = 'c';

7.         //2、直接赋值

8.         char[] arr2 = {'a','b','c'};

9.         //2、创建并初始化

10.         char[] arr3 = new char[]{'a','b','c'};

11.         System.out.println("arr1 = " + Arrays.toString(arr1));

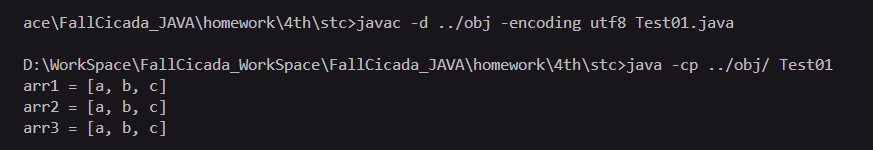
12.         System.out.println("arr2 = " + Arrays.toString(arr2));

13.         System.out.println("arr3 = " + Arrays.toString(arr3));

14.     }

15.

**运行结果：**

****

1. 有数组 String[] arr = new String[]{"hello","world","!"}; 请遍历该数组并输出每个元素

**答：**

1. public static void main(String[] args) {

2.         String[] arr = new String[]{"hello","world","!"};

3.         for(String str : arr){

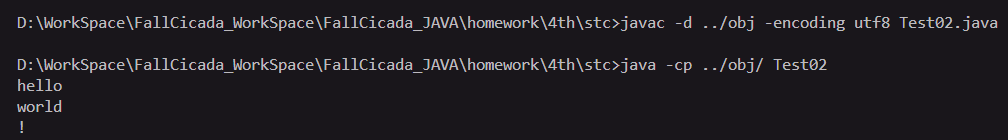
4.             System.out.println(str);

5.         }

6.     }

7.

**运行结果：**

****

1. 求一个int[]数组的次大值

根据数组的第一个和第二个元素的大小关系确定最大值和次大值从，第三个元素开始依次比较和最大值、次大值的大小关系，确定是否需要改变最大值和次大值的值

1. public class Test03 {

2.     //根据数组的第一个和第二个元素的大小关系确定最大值和次大值从，第三个元素开始依次比较和最大值、次大值的大小关系，确定是否需要改变最大值和次大值的值

3.     public static void main(String[] args) {

4.         int[] a = new int[]{1,2,3,4};

5.         secondMax(a);

6.     }

7.     static void secondMax(int[] arr){

8.         // 初始化最大值和次大值

9.         int max = 0;

10.         int secondMax = 0;

11.         // 检查数组长度

12.         if (arr.length < 2) {

13.             System.out.println("数组至少需要两个元素");

14.             return;

15.         }

16.         // 首先确定前两个元素的最大值和次大值

17.         if (arr[0] > arr[1]) {

18.             max = arr[0];

19.             secondMax = arr[1];

20.         } else {

21.             max = arr[1];

22.             secondMax = arr[0];

23.         }

24.         // 从第三个元素开始遍历

25.         for (int i = 2; i < arr.length; i++) {

26.             if (arr[i] > max) {

27.                 secondMax = max;

28.                 max = arr[i];

29.             } else if (arr[i] > secondMax && arr[i] != max) {

30.                 secondMax = arr[i];

31.             }

32.         }

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

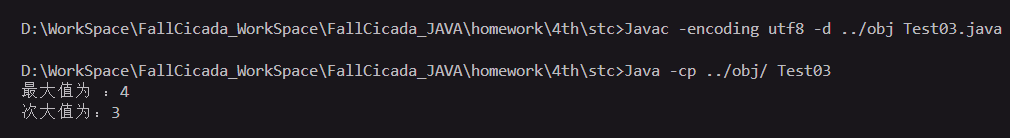
34.         System.out.println("次大值为：" + secondMax);

35.     }

36. }

37.

**运行结果：**



1. 数组内存考核：分析以下代码的运行结果，并画出代码对应的内存图

public class Test05\_memory {

public static void main(String[] args) {

int[] arr1 = {1,2,3};

int[] arr2 = arr1;

arr2[1] = 0;

System.out.println("arr1: " + Arrays.toString(arr1));

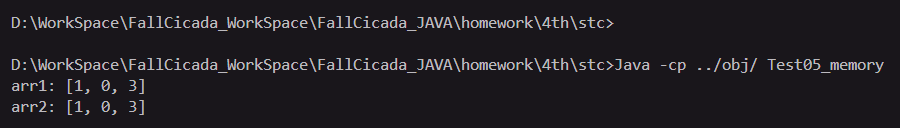
System.out.println("arr2: " + Arrays.toString(arr2));

}

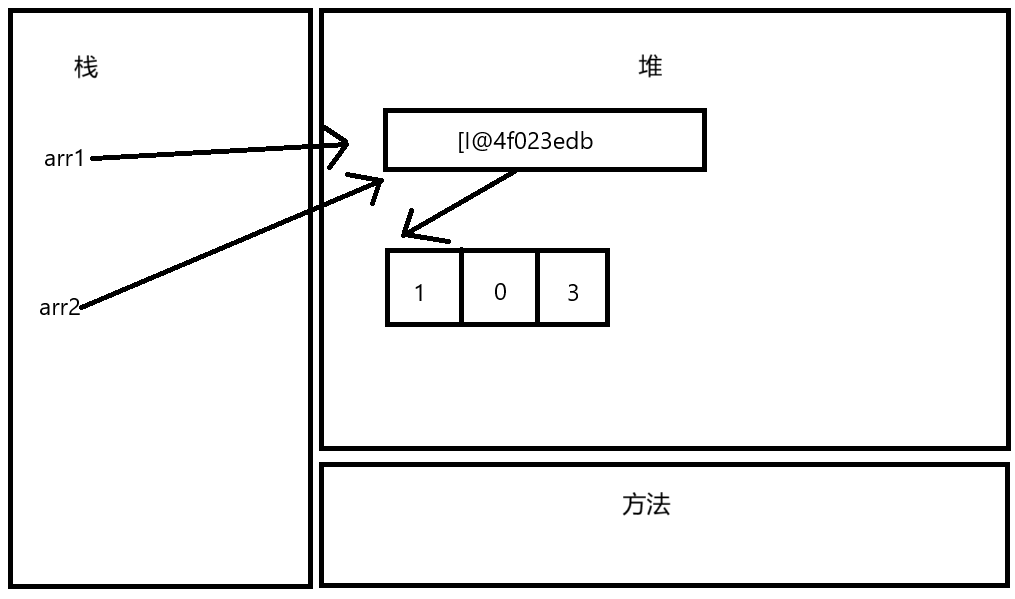
}

**答：**

定义数组arr1并赋值1、2、3，再定义数组arr2，将arr1的内存引用赋给arr2.给arr2[1]赋值0，因两数组的引用相同故两数组的索引1元素同时更改，故：



**内存图：**

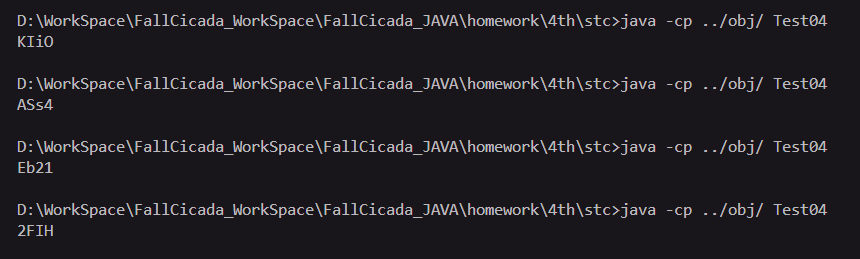
****

1. 生成验证码：随机产生一个长度为4位的验证码，包含大小写字母以及数字。 提示：使用随机数和数组。

**答：**

* 1. public class Test04 {
  2. public static void main(String[] args) {
  3. System.out.println(randomCode(4));
  4. }
  5. static String randomCode(int length) {
  6. // 使用StringBuilder来构建最终的随机代码字符串
  7. StringBuilder code = new StringBuilder();
  8. // 创建一个随机数生成器
  9. Random random = new Random();
  10. // 定义包含所有可能字符的字符集
  11. String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
  12. // 循环指定次数以达到所需的随机代码长度
  13. for (int i = 0; i < length; i++) {
  14. // 随机选择字符集中的一个字符的索引位置
  15. int index = random.nextInt(chars.length());
  16. // 将随机选中的字符添加到代码字符串中
  17. code.append(chars.charAt(index));
  18. }
  19. // 返回构建好的随机代码字符串
  20. return code.toString();
  21. }
  22. }

**运行结果:**



1. 利用随机数生成生成一个长度为20的正整型数组。然后等待用户输入要查找的数字。

1. 如果数组中包含用户输入数字，提示用户数字所在数组下标位置

2. 如果数组中不包含用户输入数字，提示用户数据不存在

3. 用户可以一直进行输入数字，如果用户输入-1，程序停止运行

**答：**

1. import java.util.Scanner;

2. import java.util.Arrays;

3.

4. public class Test06 {

5.     public static void main(String[] args) {

6.         int[] arr = new int[20];

7.         for (int i = 0; i < arr.length; i++) {

8.             arr[i] = (int) (Math.random() \* 100);

9.         }

10.         System.out.println("数组为：" + Arrays.toString(arr));

11.         findBeta(arr);

12.     }

13.

14.     static void findBeta(int[] arr) {

15.         try (Scanner sc = new Scanner(System.in)) {

16.             while (true) {

17.                 System.out.println("请输入一个数字：");

18.                 int num = sc.nextInt();

19.                 if (num == -1) {

20.                     System.out.println("退出程序");

21.                     break;

22.                 }

23.                 boolean found = false;

24.                 for (int i = 0; i < arr.length && !found; i++) {

25.                     if (arr[i] == num) {

26.                         System.out.println("数据存在, 索引为：" + i);

27.                         found = true;

28.                     }

29.                 }

30.                 if (!found) {

31.                     System.out.println("数据不存在");

32.                 }

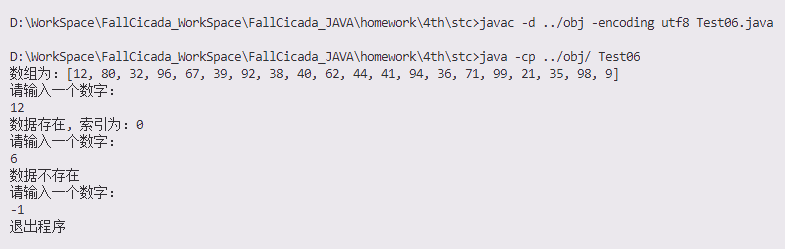
33.             }

34.         }

35.     }

36. }

**运行结果：**

****

1. 统计次数

1. 产生100个[1,6]之间的随机数，并统计每个数字出现的概率

2. 可以创建2个数组分别保存随机数信息和统计产生次数信息

3. 例如：随机数生成了10个1，20个2，30个3，5个4，10个5，25个6，对应的

统计数组内容[10,20,30,5,10,25]，即统计数组的每一个位置上存放的是对应

下标出现的次数

**答：**

1. import java.util.Random;

2. import java.util.Arrays;

3. public class Test07 {

4.     public static void main(String[] args) {

5.         // 创建一个数组用于保存产生的随机数

6.         int[] randomNumbers = new int[100];

7.         // 创建一个数组用于统计每个数字出现的次数

8.         int[] countArray = new int[6];

9.

10.         // 生成100个[1,6]之间的随机数并保存到randomNumbers数组中

11.         Random random = new Random();

12.         for (int i = 0; i < randomNumbers.length; i++) {

13.             randomNumbers[i] = random.nextInt(6) + 1;

14.             // 更新countArray中对应下标的计数值

15.             countArray[randomNumbers[i] - 1]++;

16.         }

17.         System.out.println("生成数组：" + Arrays.toString(randomNumbers));

18.         // 打印统计结果

19.         for (int i = 0; i < countArray.length; i++) {

20.             System.out.println("数字 " + (i + 1) + " 出现次数: " + countArray[i]);

21.         }

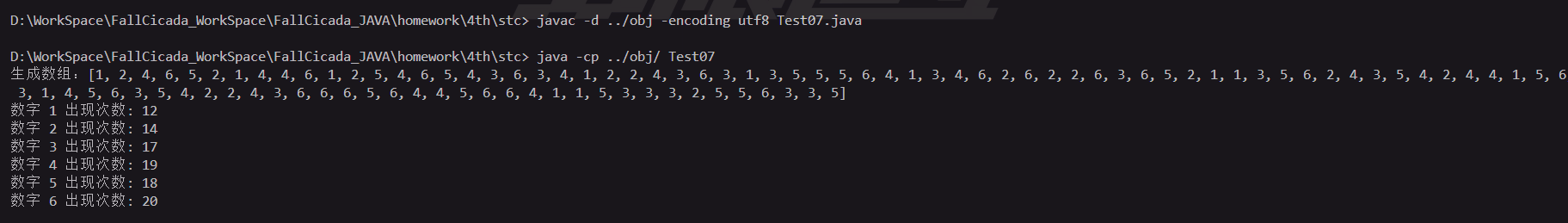
22.     }

23.

24. }

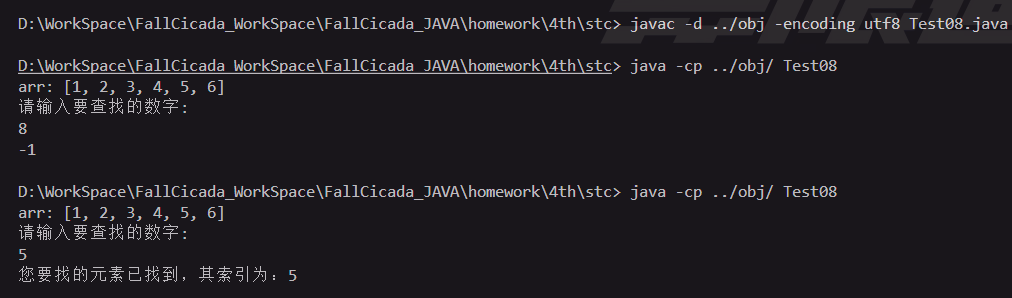
25.

**运行结果：**

****

1. 给定一个整型数组和一个目标值，要求编写一个函数来查找这个值在数组中的索引。如果找到，则返回索引；如果未找到，则返回-1。
   1. import java.util.Scanner;
   2. import java.util.Arrays;
   3. public class Test08 {
   4. public static void main(String[] args) {
   5. int[] arr = {1, 2, 3, 4, 5, 6};
   6. System.out.println("arr: " + Arrays.toString(arr));
   7. Scanner scanner = new Scanner(System.in);
   8. System.out.println("请输入要查找的数字: ");
   9. int num = scanner.nextInt();
   10. int index = findIndex(arr, num);
   11. if (index != -1) {
   12. System.out.println("您要找的元素已找到，其索引为：" + (index + 1));
   13. } else {
   14. System.out.println("-1");
   15. }
   16. }
   18. static int findIndex(int[] arr, int num) {
   19. for (int i = 0; i < arr.length; i++) {
   20. if (arr[i] == num) {
   21. return i;
   22. }
   23. }
   24. return -1;
   25. }
   26. }

**运行结果：**



1. 查找数组中第一个大于给定值的元素

给定一个整型数组和一个目标值，要求编写一个函数来查找数组中第一个大于给定值的元素。如果找到，则返回该元素的索引；如果未找到，则返回-1。

1. import java.util.Arrays;

2. import java.util.Scanner;

3.

4. public class Test09 {

5.     public static void main(String[] args) {

6.         int[] arr = {1, 2, 3, 4, 5, 6};

7.         System.out.println("arr: " + Arrays.toString(arr));

8.         Scanner scanner = new Scanner(System.in);

9.         System.out.println("请输入要查找的数字: ");

10.         int num = scanner.nextInt();

11.

12.         int index = findFirstBata(arr, num);

13.

14.         if (index != -1) {

15.             System.out.println("找到的第一个大于 " + num + " 的元素的索引为：" + (index + 1));

16.         } else {

17.             System.out.println("-1");

18.         }

19.     }

20.

21.     static int findFirstBata(int[] arr, int num) {

22.         for (int i = 0; i < arr.length; i++) {

23.             if (arr[i] > num) {

24.                 return i;

25.             }

26.         }

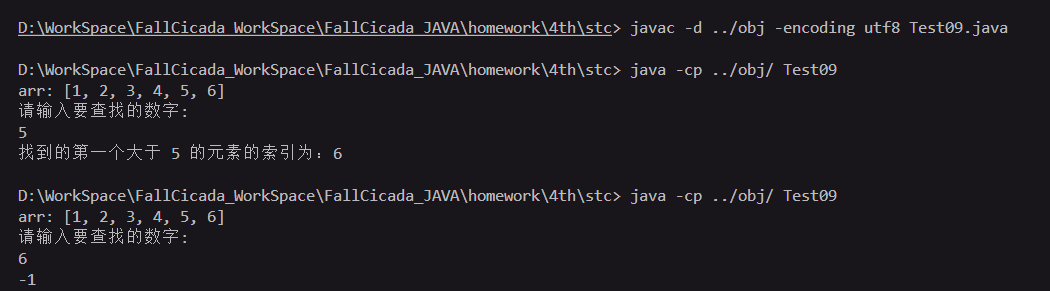
27.         return -1;

28.     }

29. }

30.

**运行结果：**



1. 描述什么情况下会出现空指针异常和索引越界异常并写代码验证。

**答：**

1. public class Test10 {

2.     // 忽略与未检查的空类型相关的警告

3.     @SuppressWarnings("null")

4.     // 主函数入口，无参数，无返回值

5.     public static void main(String[] args) {

6.         arrayIndexOut();

7.     }

8.     static void arrayIndexOut(){

9.         // 初始化一个整型数组

10.         int[] arr = {1, 2, 3};

11.         // 输出数组越界的元素，展示可能的运行时异常j

12.         System.out.println(arr[3]);

13.     }

14.     static void nullPointerException(){

15.         // 声明一个字符串变量并赋值为 null

16.         String str = null;

17.         // 尝试调用 null 字符串的长度方法，展示可能的空指针异常

18.         System.out.println(str.length());

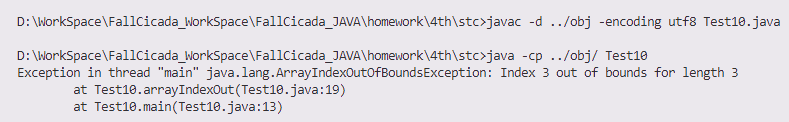
19.     }

20. }

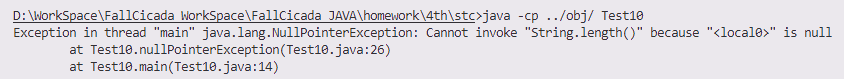
21.

**运行结果：**

**索引溢出：**

****

**空指针异常：**

****