福建師範大學
FUJIAN NORMAL UNIVERSITY

# Java语言

学　　期：2025-2026-1

教　　师：赵珊珊

# 第1-3章作业

# 第1章

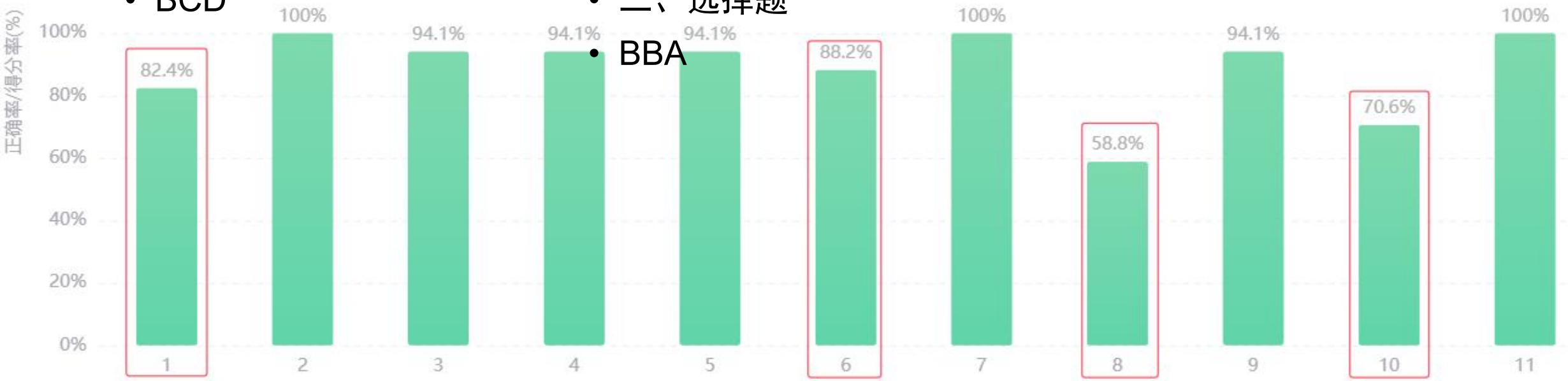
- 一、填空题
- 1. JavaSE JavaEE JavaME
- 2. JRE
- 3. java class
- 二、选择题
- BCD

# 第2章

- 一、填空题
- 1. 基本数据类型 引用数据类型
- 2. 1
- 3. 强制类型转换
- 4. 10
- 二、选择题
- BBA

# 第3章

- 一、选择题
- CCBB
- 二、填空题
- 1. 数据类型
- 2. 索引



# 错误类型

拼写错误

Jave SE

JAVA MM

答案不完整

强制

for循环

```
for(int i=0,j=1;j<5;j+=3) {
    i=i+j;
    System.out.println("i="+i+" j="+j);
}
```

```
i=1 j=1
i=5 j=4
```

数据类型与运算符

```
8.1f%5 8.1d%5
3.1000004
3.0999999999999996
```

后自减/加

```
int k=10;
if(6>8||9<k--) {
    k++;
}
else
    k--;
```

先使用k的当前值10进行9<10的比较（结果为true），再将k减1变为9；然后k++，先使用了k的值（9），然后再执行k = k + 1（k变回10）

前自减/加

```
int k=10;
if(6>8||9<--k) {
    k++;
}
else
    k--;
```

先对k的当前值10减1得到9，再进行9<9的比较（结果为false）；然后k--，先使用了k的值（9），然后再执行k = k - 1（k变为8）

数组的栈地址

数组的引用是一个变量，存放**数组对象在堆内存中的首地址**；

栈地址是数组引用变量在*栈内存*中的位置地址，可以看成是保管引用变量的抽屉的编号；

栈地址通常由系统管理，程序员一般不直接操作或获取；

Java编程中，开发者是无法直接获取或操作引用变量自身的栈地址的；

Java语言刻意隐藏了这部分细节，开发者只需关心引用变量及其指向的堆对象。

# 第3章

```
请输入一个各位数字不全相同的四位数: 8
错误！不是四位数，请重新输入：
88888
错误！不是四位数，请重新输入：
8888
错误！每位数字不能完全相同，请重新输入：
6888
开始验证卡普耶卡变换（6174问题）：
第1步: 8886 - 6888 = 1998
第2步: 9981 - 1899 = 8082
第3步: 8820 - 288 = 8532
第4步: 8532 - 2358 = 6174
经过 4 步变换后，达到卡普耶卡常数 6174。
```

```java
import java.util.Arrays;
import java.util.Scanner;

public class Verify6174 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("请输入一个各位数字不全相同的四位数: ");
        int inputNumber = scanner.nextInt();

        // 验证输入是否为四位数且不全相同
        while (inputNumber < 1000 || inputNumber > 9999) {
            System.out.println("错误！不是四位数，请重新输入:");
            inputNumber = scanner.nextInt();
        }
        while (isAllDigitsSame(inputNumber)) {
            System.out.println("错误！每位数字不能完全相同，请重新输入：");
            inputNumber = scanner.nextInt();
        }
        scanner.close();
        int currentNumber = inputNumber;
        int step = 0;
        System.out.println("开始验证卡普耶卡变换（6174问题）:");

        // 循环直到得到6174
        while (currentNumber != 6174) {
            step++;
            int[] digits = getDigits(currentNumber);
            Arrays.sort(digits); // 升序排序得到最小数的基础
            int min = formNumber(digits, false); // 升序排列构成最小数
            int max = formNumber(digits, true);  // 降序排列构成最大数
            currentNumber = max - min;
            System.out.printf("第%d步: %d - %d = %d%n", step, max, min, currentNumber);
        }
        System.out.printf("经过 %d 步变换后，达到卡普耶卡常数 6174。\n", step);
    }

    private static boolean isAllDigitsSame(int number) {
        int digit1 = number % 10;
        int digit2 = (number / 10) % 10;
        int digit3 = (number / 100) % 10;
        int digit4 = number / 1000;
        return digit1 == digit2 && digit2 == digit3 && digit3 == digit4;
    }

    private static int[] getDigits(int number) {
        int[] digits = new int[4];
        digits[0] = number % 10;          // 个位
        digits[1] = (number / 10) % 10;   // 十位
        digits[2] = (number / 100) % 10;  // 百位
        digits[3] = number / 1000;        // 千位
        return digits;
    }

    // 根据提供的数字数组和排序方向构造数字
    private static int formNumber(int[] digits, boolean descending) {
        int number = 0;
        if (descending) {
            // 降序构造最大数：从最高位（千位）到最低位（个位）依次放置最大的数字
            for (int i = 3; i >= 0; i--)
                number = number * 10 + digits[i];
        } else {
            // 升序构造最小数：从最高位到最低位依次放置最小的数字
            for (int i = 0; i < 4; i++)
                number = number * 10 + digits[i];
        }
        return number;
    }
}
```