语言基础

1 学习目标

学习和熟悉Java语言的基本语法：
（1）熟悉基本的语言符号
（2）掌握值与变量的用法
（3）掌握控制语句的用法
（4）掌握函数的声明与调用
（5）熟悉标准输入与输出的用法

2 案例学习

通过对一些简单例子的学习，理解Java语言的基本语法。

2.1 语言符号

（1）保留字（关键词）是计算机语言预定义的有特定意义的符号，不能作为标识符（如变量名、类名等）使用。保留字通常用于控制程序的结构或功能。常见保留字：

控制结构：`if`, `else`, `switch`, `while`, `for`, `do`
定义数据类型：`int`, `float`, `char`, `boolean`, `double`, `class`, `interface`
其他：`public`, `private`, `protected`, `static`, `final`, `return`, `void`, `new`, `this`

```
// 保留字：public、class、static、void、int
public class Main {
  public static void main(String[] args) {
    int x = 10;
  }
}
```

（2）运算符（操作符）用于在表达式中对变量或值进行操作。Java语言提供了多种运算符：

算术运算符：`+`, `-`, `*`, `/`, `%`
赋值运算符：`=`, `+=`, `-=`, `*=`, `/=`
比较运算符：`==`, `!=`, `>`, `<`, `>=`, `<=`
逻辑运算符：`&&`, `||`, `!`
位运算符：`&`, `|`, `^`, `~`, `<<`, `>>`, `>>>`
三元运算符（三目运算）：`condition ? value1 : value2`

```
public class Main {
  public static void main(String[] args) {
    int a = 10, b = 20; // 赋值操作
    System.out.println(a + b); // 加法运算符
    System.out.println(a > b); // 比较运算符
    System.out.println(a == b && a < b); // 逻辑运算符
  }
}
```

（3）标识符用于命名变量、方法、类等。标识符必须以字母、下划线`_`或美元符号`$`开头，后面可以跟字母、数字、`_`或`$`，但不能使用保留字作为标识符。

```
// 标识符：myVariable、MyClass、main、args、System、out、println、String
public class MyClass {  // MyClass是类名标识符
  public static void main(String[] args) {
    int myVariable = 10;  // myVariable是变量名标识符
    System.out.println(myVariable);
  }
}
```

（4）分隔符用来分隔语句、代码块等：

```
括号：(), {}, []
分号：; 表示语句结束
逗号：，用于分隔多个变量或参数
点号：. 用于访问成员
```

（5）字面量也是一种符号，表示固定值或常量。

```
int age = 10; // 整型
long numberOfstars = 123456789L; // 长整型
double pi = 3.14159; // 双精度
float weight = 9.8f; // 浮点数
char letter = 'A'; // 字符
boolean isJavaFun = true; // 布尔值
String message = "Hello, World!"; // 字符串
Object obj = null; // 空引用
```

2.2 值与变量

（1）值表示被程序操作的一个数据实体，例如一个数字或一个字符等。值的类型分为两种，一种是原始类型（Primitive Types），另一种是引用类型（Reference Types）。

（2）变量是对内存的抽象，可以看作存储特定类型值的内存盒子，变量通过变量名来引用。Java是静态强类型语言：静态表示变量类型在编译阶段确定；强类型表示变量支持的操作受限于其类型。

```
int x = 10;
double y = 10.0;
x = x >> 1;
y = y >> 1; // 错误：浮点数不能进行右移操作
```

2.3 控制语句

（1）转换分数为等级。

```
public class Test {
  public static void main(String[] args) {
    int score = 59;
    if(score >= 90) {
      System.out.println("A");
    }else if(score >= 80){
      System.out.println("B");
    }else if(score >= 70) {
      System.out.println("C");
    }else if(score >= 60) {
      System.out.println("D");
    }else{
      System.out.println("F");
    }
  }
}
```

（2）数字转换成季节。

```
public class Test {
  public static void main(String[] args) {
    int season = 4;
    switch(season) {
      case 1: System.out.println("春季");break;
      case 2: System.out.println("夏季");break;
      case 3: System.out.println("秋季");break;
      case 4: System.out.println("春季");break;
      default: System.out.println("未知");
    }
  }
}
```

(3) 计算`1+2+3+...+n`。

```
public class Test {
  public static void main(String[] args) {
    int n = 4;
    int sum = 0; // 存放求和结果
    for(int i = 1; i <= n; i++) {
      sum += i;
    }
    System.out.println("结果: " + sum);
  }
}
```

(4) 通过迭代的方式对数字`n`进行随机减少操作，直到为零。

```
public class Test {
  public static void main(String[] args) {
    int n = 400;
    while(n > 0) {
      n -= (int)(Math.random() * 10); // 随机减少一个整数
      if(n < 0) {
        n = 0; // 若为负数，则修正为0
      }
      System.out.printf("%-4d", n); // 打印当前值
    }
  }
}
```

第二种方式：

```
public class Test {
  public static void main(String[] args) {
    int n = 400;
    while(true) {
      n -= (int)(Math.random() * 10); // 随机减少一个整数
      if(n < 0) {
        n = 0; // 若为负数，则修正为0
      }
      System.out.printf("%-4d", n); // 打印当前值
      if(n == 0) break; // 结束循环
    }
  }
}
```

(5) 打印偶数。

```
public class Test {
  public static void main(String[] args) {
    int n = 20;
    for(int i = 0; i < n; i++) {
      if(i % 2 == 0)
        System.out.printf("%-3d", i);
    }
  }
}
```

另一种方式：

```
public class Test {
  public static void main(String[] args) {
    int n = 20;
    for(int i = 0; i < n; i++) {
      if(i % 2 != 0) continue; // 结束当前迭代
      System.out.printf("%-3d", i);
    }
  }
}
```

2.4 函数

（1）熟悉函数的声明、定义和调用，包括修饰符、返回类型、函数签名、函数体等。

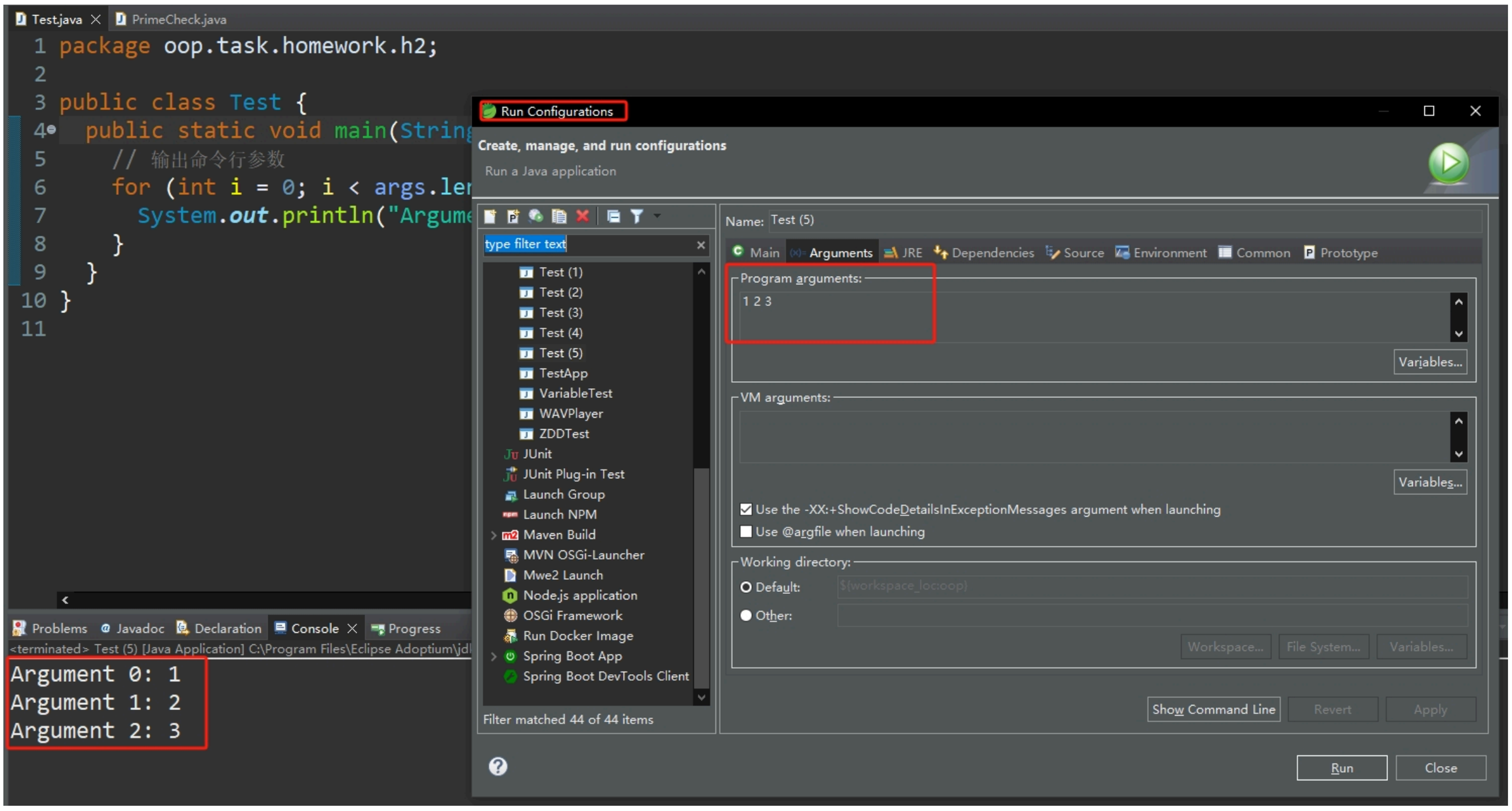
（2）通过一个例子来熟悉函数的定义和使用。假设需要筛选出小于`maxN`的全部素数，具体计算过程封装在两个函数中，`isPrime()`用于判断当前数是否为素数，`printPrimes()`用于打印所有小于`maxN`的素数。

```
public class PrimeCheck {
  public static void main(String[] args) {
    int maxN = 100;
    printPrimes(maxN);
  }
  public static void printPrimes(int maxN) {
    for(int i = 1; i < maxN; i++) {
      if(isPrime(i)) {
        System.out.printf("%-3d", i);
      }
    }
  }
  public static boolean isPrime(int number) {
    if (number <= 1) return false; // 1及以下的数都不是素数
    for (int i = 2; i <= Math.sqrt(number); i++) { // 只需检查到平方根
      if (number % i == 0) return false; // 如果能被整除，则不是素数
    }
    return true; // 否则是素数
  }
}
```

（3）入口函数参数的作用。入口函数`main()`的参数`args`用于在运行程序时接收外部传入的命令行参数。

```
public class Example {
  public static void main(String[] args) {
    // 输出命令行参数
    for (int i = 0; i < args.length; i++) {
        System.out.println("Argument " + i + ": " + args[i]);
    }
  }
}
```

若在Eclipse中运行，则通过`Run Configurations`窗口配置参数，在`Arguments`选项卡中输入参数。



若在命令行中运行，则直接在命令后输入参数。

```
D:\works\oop\bin>
D:\works\oop\bin>
D:\works\oop\bin>java oop.task.homework.h2.Test 1 2 3
Argument 0: 1
Argument 1: 2
Argument 2: 3

D:\works\oop\bin>
```

2.5 标准IO

（1）通过`System.out`来使用标准输出流。

（2）通过`Scanner`和`System.in`来使用标准输入流。

`Scanner`的next()系列函数是阻塞式的，会一直等待标准输入流（`System.in`）中有数据可供读取。一般情况下，用户在控制台输入数据并按下回车键后，数据会被传输到标准输入流（`System.in`）。

```
public class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.printf("请输入姓名:");
    String name = input.nextLine(); // 阻塞，等待标准输入中有数据
    System.out.printf("你好，%s", name);
    input.close();
  }
}
```

2.6 综合案例

编写一个命令行播放器程序，支持WAV格式音频文件的播放，支持简单的歌曲选择功能。

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class WAVPlayer {
  private Clip audioClip; // 音频控制对象

  // 播放wav文件
  public void play(File wav) {
    // 创建一个新线程来播放音频，避免阻塞主线程
    new Thread(() -> {
      try(AudioInputStream audioStream = AudioSystem.getAudioInputStream(wav);) { // 获取音频文件
        // 如果已经有音频在播放，先停止并释放资源
        if (audioClip != null && audioClip.isRunning()) {
          audioClip.stop();
          audioClip.close();
        }
        // 打开并开始播放音频
        audioClip = AudioSystem.getClip();
        audioClip.open(audioStream);
        audioClip.start(); // 非阻塞方式
        System.out.printf("准备播放:%s\n", wav.getName());
        try {
          Thread.sleep(1000); // 应对资源加载延时
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        // 等待音频播放完毕
        audioClip.drain();
        audioClip.close();
      } catch (UnsupportedAudioFileException | IOException | LineUnavailableException ex) {
        System.out.println("播放错误");
      }
    }).start();  // 启动新线程
}
public void stop() {
  if (audioClip != null && audioClip.isRunning()) {
    audioClip.stop();
    audioClip.close();
  }
}
public static void main(String[] args) {
  WAVPlayer player = new WAVPlayer(); // 实例化播放器
  try(Scanner input = new Scanner(System.in)){
    System.out.print("文件目录: ");
    String dir = input.nextLine().trim(); // 读入输入的目录
    File file = new File(dir);
    List<File> wavs = new ArrayList<File>();
    for(File f : file.listFiles()) {
      if(f.getName().toLowerCase().contains(".wav")) {
        wavs.add(f);
      }
    }
    System.out.println("歌曲目录:");
    for(int i = 0; i < wavs.size(); i++) {
      System.out.printf("%d.%s\n", i + 1, wavs.get(i).getName());
    }
    while(true) {
      String number = input.nextLine().trim(); // 读取曲目序号
      int index = -1;
```

```
        try {
          index = Integer.parseInt(number) - 1;
        }catch(NumberFormatException ex) {
          System.out.println("歌曲序号格式错误"); // 序号格式错误
        }
        if(index < 0 || index >= wavs.size()) continue; // 索引超出范围
        File target = wavs.get(index);
        player.play(target);
      }
    }
  }
}
```

3 作业任务

3.1 计算器

设计一个计算器，支持双精度值的四则运算。要求输入的是两个数值，以及运算符。

```
public class SimpleCalculator {
  // 计算方法，根据操作符进行相应的计算
  public static double calculate(double num1, double num2, String operator) {
    // 逻辑实现：运算符解析与计算过程
  }

  // 主方法
  public static void main(String[] args) {
    if(args.length != 3) {
      System.out.println("输入错误");
    }else {
      double v1 = Double.parseDouble(args[0]);
      double v2 = Double.parseDouble(args[2]);
      String operator = args[1];
      System.out.println(calculate(v1, v2, operator));
    }
  }
}
```

3.2 回文数识别

设计一个回文数识别函数，支持对回文数进行判断。回文数的定义是正读和倒读都一样的数。

```
public class PalindromeNumbers {
  // 检查一个数是否为回文数的方法
  public static boolean isPalindrome(int number) {
    // 逻辑实现
  }
  // 反转数字
  public static int reverse(int number) {
    // 逻辑实现
  }

  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    System.out.printf("从1到%-3d的回文数如下:\n", n);
    for (int i = 1; i < n; i++) {
      if (isPalindrome(i)) {
        System.out.print(i + " ");
      }
    }
  }
}
```

4 作业要求

（1）类名、方法名、内部结构与任务中给出的一致。
（2）功能正确、逻辑清晰、代码规范、注释准确。
（3）在FTP服务器查找自己学号对应的文件夹，将作业对应的源代码 `src` 文件夹上传到 `/学号/作业2/` 目录下。

5 视频资源

地址：`https://www.bilibili.com/video/BV1Q7pEeXEVC/?vd_source=ab9767f8330b9092bf0b35e3238af895`