

华东师范大学数据科学与工程学院实验报告

课程名称：计算机网络与编程

年级：2020 级

上机实践成绩：

指导教师：张召

姓名：张熙翔

学号：10205501427

上机实践名称：Java 基础

上机实践日期：2022/3/15

一、实验目的

熟悉 Java 开发环境，熟悉掌握 IntelliJ IDEA 的使用，学习并掌握 Java 编程基础和面向对象部分基础知识，为之后使用 Java 进行网络编程打下基础。

二、实验任务

Java 开发环境配置。

熟悉变量、运算符、控制流程、数组、字符串、I/O 以及类和对象。

熟悉继承、多态、接口、抽象类、异常处理以及枚举。

三、使用环境

IntelliJ IDEA Version 2021.3.2

四、实验过程

week1_task1:

创建一个以自己学号+姓名命名的 java 项目，编写 java 代码以实现将自己的学号和姓名打印出来的效果。



The screenshot displays the IntelliJ IDEA interface. The top pane shows a Java file named '张熙翔10205501427.java' with the following code:

```
1 package Test;
2
3 public class 张熙翔10205501427 {
4     public static void main(String[] args) {
5         System.out.println("10205501427 张熙翔");
6     }
7 }
```

The bottom pane shows the '运行' (Run) window for the class '张熙翔10205501427'. It displays the command executed: 'D:\JAVA\bin\java.exe "-javaagent:D:\J' and the output: '10205501427 张熙翔'. The status indicates '进程已结束,退出代码0' (Process ended, exit code 0).

week2_task1:

```
char c1= '中' ;//正确
char c2= '哈哈' ;//错误 字符数过多
float f1= 54.321 ;//错误 默认是 double 型
boolean bl1= 1 ;//错误 取值只有 true 和 false
```

week2_task2:

返回 a 和 b 的最大公约数。

week2_task3:

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

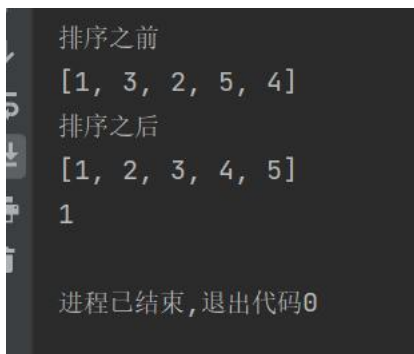


```
*
**
***
****
*****

进程已结束,退出代码0
```

week2_task4:

```
int[] a = new int[]{1, 3, 2, 5, 4};
System.out.println("排序之前");
System.out.println(Arrays.toString(a));
Arrays.sort(a);
System.out.println("排序之后");
System.out.println(Arrays.toString(a));
System.out.println(Arrays.binarySearch(a, 2));
```



```
排序之前
[1, 3, 2, 5, 4]
排序之后
[1, 2, 3, 4, 5]
1

进程已结束,退出代码0
```

week2_附加题:

int 是基本数据类型, Integer 是引用数据类型; Integer 是 int 的包装类。要想改变默认的排列顺序,不能使用基本类型,而要使用它们对应的包装类。

week2_task5:

`equals()`判断两个变量或者实例指向同一个内存空间的值是不是相同;

而`==`是判断两个变量或者实例是不是指向同一个内存空间。

```

true
false
true
true
true

```

```

String str1 = "the light";
String str2 = str1;
String str3 = new String(str1);
String str4 = "the light";
String str5 = "the " + "light";
System.out.println(str1 == str2);
//true 创建 str2 的时候, str2 指向了 str1, 所以 str1 和 str2 指向同一个地址, 存储的数据自然相同
System.out.println(str1 == str3);
//false String 类型属于引用类型, 变量中存储的是地址, str3 重新 new 了一个类 与 str1 地址不同
System.out.println(str1 == str4);
//true "the light" 这样的数据只会存储一个地址, 所以它们的引用都是指向的同一块地址
System.out.println(str1 == str5);
//true "the light" 这样的数据只会存储一个地址, 所以它们的引用都是指向的同一块地址
System.out.println(str1.equals(str4));
//true str1 存储地址指向的数据和 str4 存储的地址指向的数据都是"the light"

```

week2_task6:

一般认为, Java 对象都是在堆上分配的, 但也有一些特殊情况会出现栈上分配、TLAB 分配、年老代分配。

Java 所有类都具有的方法: `wait`, `notify`, `notifyAll`, `equals`, `hashCode`, `toString`, `clone`, `finalize`。

week2_task7:

`private` 修饰: 不正确。因为 `private` 表示私有, 就是除了 `OuterClass` 这个类之外, 任何其他的不可以直接使用。

`protected` 修饰: 正确。因为对于朋友类来说, `InnerClass` 是 `public` 的, 没有任何限制, 可以使用。

week2_附加题:

不正确。`OuterClass` 不是封闭类。`InnerClass` 不是静态的, 因此它需要外部类的实例。

正确的形式: 将 `class InnerClass` 改为 `static class InnerClass`


week2_task8:

`next()`方法不能得到带空格的字符串, 对输入有效字符之前遇到的结束符会自动过滤掉; 在读取到有效字符之后, `next()`方法才将其后的空格键、`Enter` 键等视为结束符。

`nextLine()`方法可以得到带空格的字符串, 它的结束符只能是 `Enter` 键, 返回的是 `Enter` 键之前没有被读取的所有字符。

week2_task9:

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int[] a=new int[10];
    for(int i=0;i<a.length;i++){
        a[i]=in.nextInt();
    }
    Arrays.sort(a);
    PrintWriter out = new PrintWriter(System.out);
    for(int i=0;i<a.length;i++){
        out.print(a[i]+" ");
        out.flush();
    }
}
```



0 9 8 7 6 5 4 3 2 1
0 1 2 3 4 5 6 7 8 9
进程已结束,退出代码0

week3_task1:

```
package Test;
import java.lang.Thread;
import java.util.Date;

class Watch {
    public Date startTime;
    public Date endTime;

    public void start(){
        startTime = new Date();
    }
    public void stop(){
        endTime = new Date();
    }
}

class Stopwatch extends Watch {
    public long getElapsedTime() {
        return endTime.getTime() - startTime.getTime();
    }
}

public class 张熙翔 10205501427 {
    public static void main(String[] args) {
        Stopwatch sw = new Stopwatch();
        sw.start();
        try{
            Thread.sleep(10);
        }catch(Exception e){}
        sw.stop();
        System.out.print(sw.getElapsedTime());
    }
}
```

week3_task2:

```
package Test;

import java.util.Scanner;

interface Compute {
    int compute(int n, int m);
}

class Add implements Compute {
    public int compute(int n, int m){
        return n+m;
    }
}

class Minus implements Compute {
    public int compute(int n, int m){
        return n-m;
    }
}

class Mul implements Compute {
    public int compute(int n, int m){
        return n*m;
    }
}

class Div implements Compute {
    public int compute(int n, int m){
        return n/m;
    }
}

class UseCompute {
    public void UseCom(Compute com, int one, int two) {
        System.out.println(com.compute(one, two));
    }
}

public class TestComputer {
    public static void main(String []args){
        Scanner reader = new Scanner(System.in);
        int a = reader.nextInt();
        int b = reader.nextInt();
        UseCompute computer = new UseCompute();
        computer.UseCom(new Add(), a, b);
        computer.UseCom(new Minus(), a, b);
        computer.UseCom(new Mul(), a, b);
        computer.UseCom(new Div(), a, b );
    }
}
```

```
10 2
```

```
12
```

```
8
```

```
20
```

```
5
```

week3_task3:

```
package Test;
```

```
public class Employee{
    private String name;
    private int month;

    public Employee(String name, int month){
        this.name = name;
        this.month = month;
    }

    public float get_Salary(int month){
        if(month == this.month){
            System.out.println("Happy Birthday to "+this.name+ "!");
            return 100;
        }
        else{
            return 0;
        }
    }

    public static void main(String[] args) {
        Employee a[] = new Employee[3];

        a[0] = new SalariedEmployee("张三", 5, 1000);
        a[1] = new HourlyEmployee("李四", 3, 2000, 200);
        a[2] = new SalesEmployee("王五", 2, 50000, (float) 0.1);

        System.out.println(a[0].name+ "的工资为: " + a[0].get_Salary(2));
        System.out.println(a[1].name+ "的工资为: " + a[1].get_Salary(2));
        System.out.println(a[2].name+ "的工资为: " + a[2].get_Salary(2));
    }
}

class SalariedEmployee extends Employee{

    private float salary;

    public SalariedEmployee(String name, int month, float salary)
    {
        super(name, month);
        this.salary = salary;
    }

    @Override
```

```
    public float get_Salary(int month)
    {
        return salary + super.get_Salary(month);
    }
}
class HourlyEmployee extends Employee
{
    private float salary; //每小时工资
    private int hour; //每月工作的小时数

    public HourlyEmployee(String name, int month, float salary, int hour)
    {
        super(name, month);
        this.salary = salary;
        this.hour = hour;
    }

    @Override
    public float get_Salary(int month)
    {
        return salary;
    }
}

class SalesEmployee extends Employee
{
    private float sale; //月销售额
    private float bonus; //提成率
    public SalesEmployee(String name, int month, float sale, float bonus)
    {
        super(name, month);
        this.sale = sale;
        this.bonus = bonus;
    }

    @Override
    public float get_Salary(int month)
    {
        return sale * bonus + super.get_Salary(month);
    }
}
```

```
张三的工资为: 1000.0
李四的工资为: 2000.0
Happy Birthday to 王五!
王五的工资为: 5100.0

进程已结束,退出代码0
```

week3_task4:

Error 类一般是指与虚拟机相关的问题，如系统崩溃，虚拟机错误，内存空间不足，方法调用栈溢出等。对于这类错误，Java 编译器不去检查他们。对于这类错误的导致的应用程序中断，仅靠程序本身无法恢复和预防，遇到这样的错误，建议让程序终止。

Exception 类表示程序可以处理的异常，可以捕获且可能恢复。遇到这类异常，应该尽可能处理异常，使程序恢复运行，而不应该随意终止异常。

week3_task5:

ClassCastException 类型强制转换异常

ArithmeticException 算术运算异常

ArrayStoreException 向数组中存放与声明类型不兼容对象异常

IndexOutOfBoundsException 下标越界异常

NumberFormatException 数字格式异常