

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：21 级

上机实践成绩：

指导教师：张召

姓名：杨茜雅

学号：10215501435

上机实践名称：Lab03

上机实践日期：3.17-3.24

上机实践编号：

组号：

上机实践时间：2023.3.17

一、实验目的

熟悉掌握IntelliJ IDEA的使用

学习并掌握 Java面向对象部分基础知识，为使用Java 进行网络编程打下基础

二、实验任务

熟悉继承、多态、接口、抽象类、异常处理以及枚举等相关知识

三、使用环境

IntelliJ IDEA

JDK 版本: Java 19

四、实验过程

Task 1:设计一个名为 **StopWatch** (秒表) 的类，该类继承 **Watch** (表) 类

```
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 Main {
    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());
    }
}
```

```
C:\Users\86138\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=55646:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8
28
Process finished with exit code 0
```

bonus task1 (optional):

回顾C++的继承方式，其有public继承、protected继承和private继承，后两种

常用作空基类优化等技巧，然而Java只有一种继承方式 extends，这是为什么？

Java旨在简化开发流程，减少程序复杂度，单一继承方式可以降低程序开发的复杂度。Java是面向对象的编程语言：Java设计的初衷是创建一种纯粹的面向对象编程语言，为了保持纯洁，Java舍弃了C++多重继承的特性。Java只提供了单一的继承方式是因为它支持 interface（接口）的概念，interface是一种纯粹的抽象类，即方法没有实现，只有方法名和方法签名。Java的接口可以被类实现，使类获得实现接口中所有方法的能力。这种方式被称为接口继承。Java可以使用接口继承来让一个类实现多个接口，从而弥补了单继承的缺陷。

task2: 对于提供的Fish类，implements Comparable接口。初始化10个Fish对象放入数组或容器，并使用按照 size 属性从小到大排序，排序后从前往后对每个对象调用print()进行打印

```
package test;

class Fish implements Comparable<Fish> {
    private final int size;

    public Fish() {
        Random r = new Random();
        this.size = r.nextInt(100);
    }

    public int compareTo(Fish o) {
        //比较此对象与指定对象的顺序。如果该对象小于、等于或大于指定对象，则分别返回负整数、零或正整数。
        return compare(this.size, o.size);
    }

    public static int compare(int size1, int size2) {
        return (Integer.compare(size1, size2));
    }

    public String toString() {
        return "Fish" + "size = " + this.size;
    }
}

public class Main {
    public static void main(String[] args) {
```

```
Fish[] fishlist = new Fish[10];
for (int i = 0; i < 10; i++) {
    fishlist[i] = new Fish();
}
System.out.println("before sort");
for (Fish Fish1 : fishlist) {
    System.out.println(Fish1.toString());
}
Arrays.sort(fishlist);
System.out.println("After sort");
for (Fish Fish1 : fishlist) {
    System.out.println(Fish1.toString());
}
}
```

实验结果：

```
before sort
Fish size=81
Fish size=64
Fish size=82
Fish size=11
Fish size=45
Fish size=47
Fish size=89
Fish size=10
Fish size=99
Fish size=94
After sort
Fish size=10
Fish size=11
Fish size=45
Fish size=47
Fish size=64
Fish size=81
Fish size=82
Fish size=89
Fish size=94
Fish size=99
```

task3: 根据要求创建**SalesEmployee**、**HourlyEmployee**、**SalariedEmployee**三个类的对象各一个，并计算某个月这三个对象员工的工资：

```
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) {  
        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);  
    }  
}
```

```
C:\Users\86138\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:E:\Pro  
张三的工资为： 1000.0  
李四的工资为： 2000.0  
王五的工资为： 5100.0  
  
Process finished with exit code 0
```

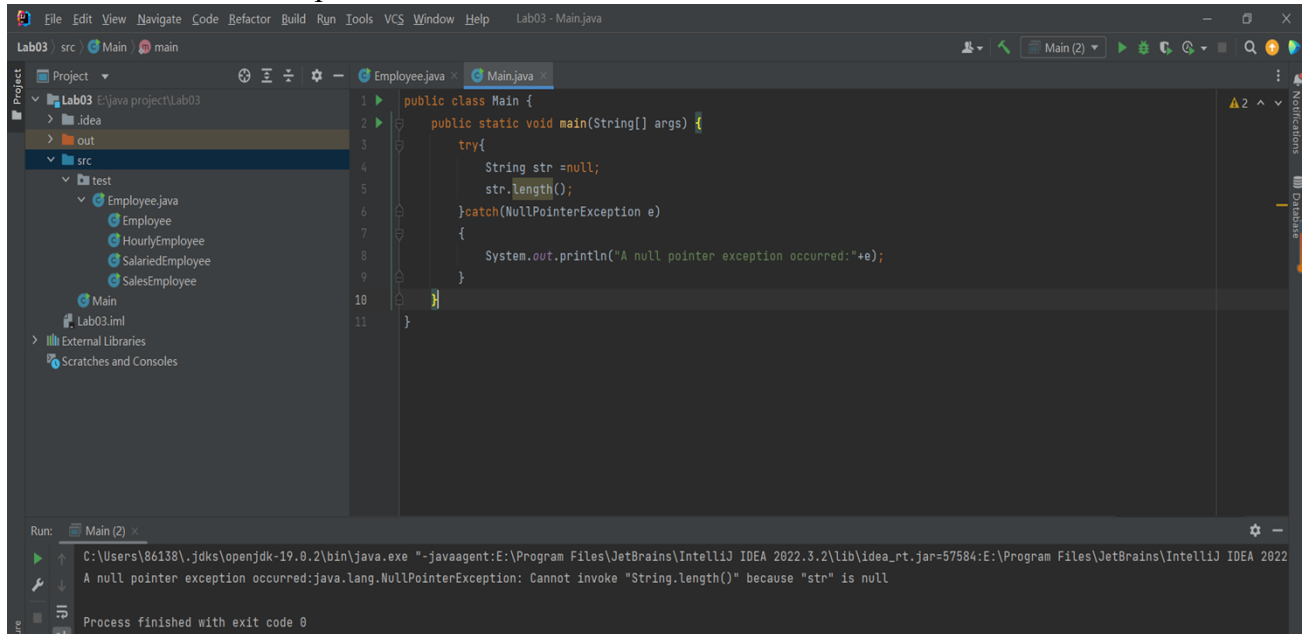
Task 4: 请在实验报告中列举出 **Error** 和 **Exception** 的区别

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。

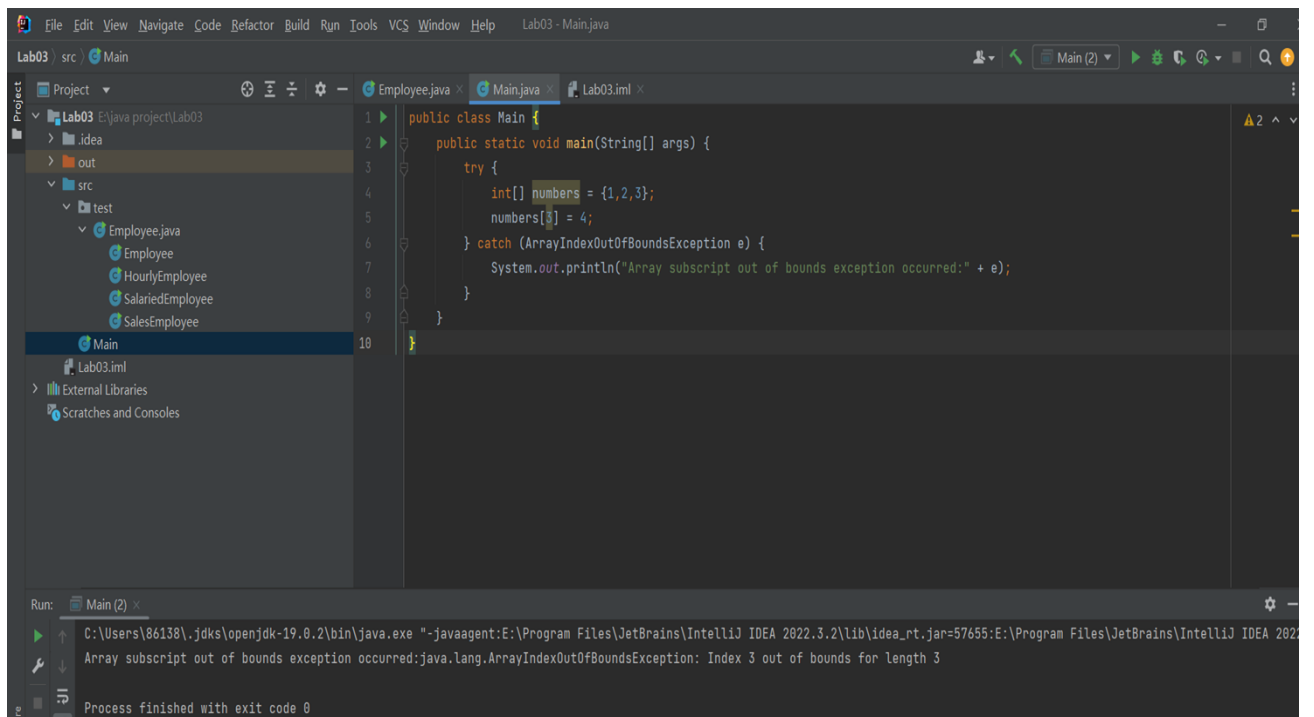
Exception 类表示程序可以处理的异常，可以捕获且可能恢复。遇到这类异常，应该尽可能处理异常，使程序恢复运行，而不应该随意终止异常。Exception 又分为可检查异常和不检查异常，可检查异常在源代码里必须显式地进行捕获处理，这是编译期检查的一部分。不检查异常就是所谓的运行时异常，类似 NullPointerException、ArrayIndexOutOfBoundsException 之类，通常是可以编码避免的逻辑错误，具体根据需要进行判断是否需要捕获，并不会在编译期强制要求。

task5: 请设计可能会发生的5个RuntimeException案例并将其捕获，将捕获成功的运行时截图和代码附在实验报告中

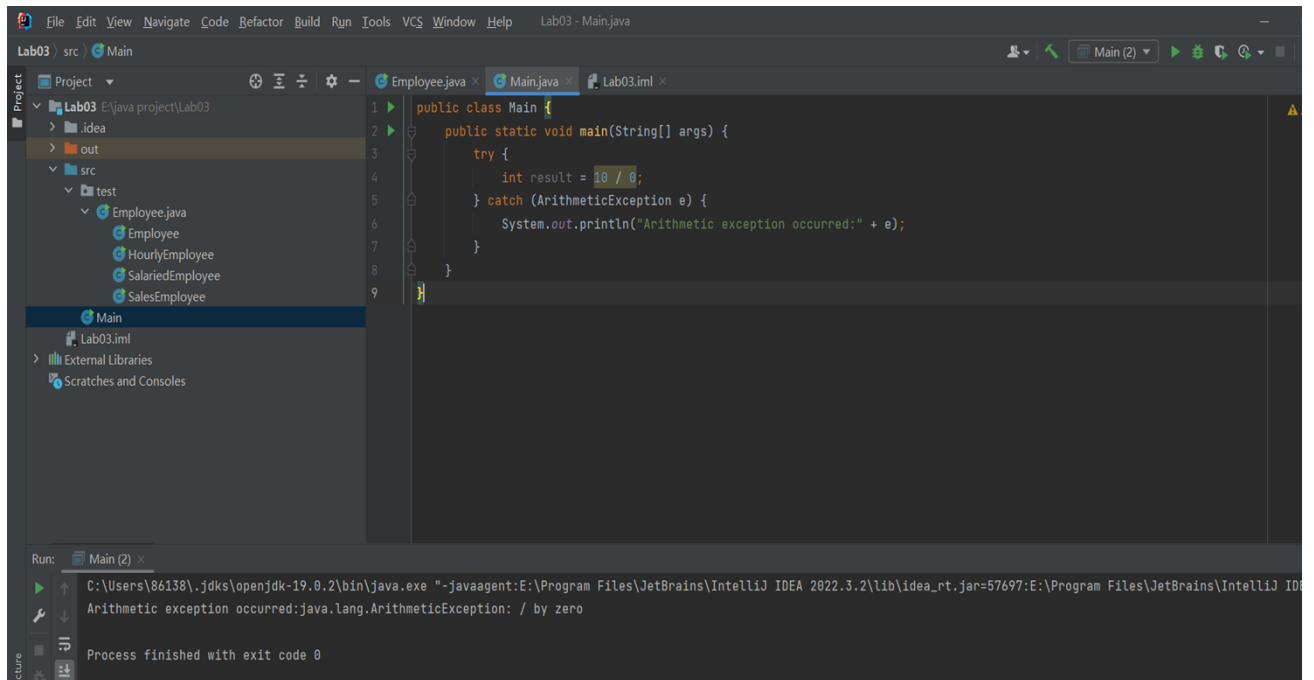
1、 NullPointerException （空指针异常）



2、 ArrayIndexOutOfBoundsException （数组下标越界异常）



3、 ArithmeticException （算术异常）



```

1 public class Main {
2     public static void main(String[] args) {
3         try {
4             int result = 10 / 0;
5         } catch (ArithmeticException e) {
6             System.out.println("Arithmetic exception occurred:" + e);
7         }
8     }
9 }

```

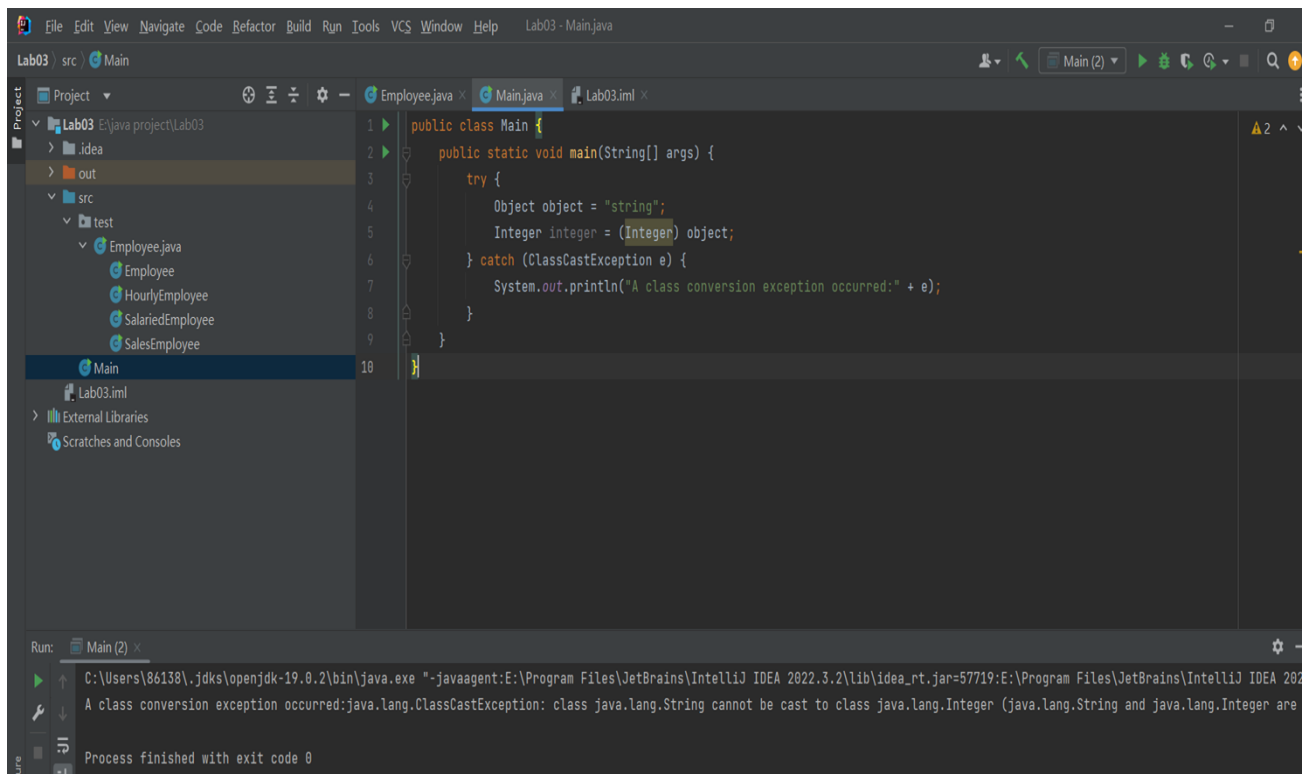
Run: Main (2) ×

C:\Users\86138\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=57697:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin" -Didea.config.path=E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\config -Didea.system.path=E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar -Didea.version=2022.3.2 -jar E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin\idea_rt.jar 57697

Arithmetic exception occurred:java.lang.ArithmeticException: / by zero

Process finished with exit code 0

4、 ClassCastException （类转换异常）



```

1 public class Main {
2     public static void main(String[] args) {
3         try {
4             Object object = "string";
5             Integer integer = (Integer) object;
6         } catch (ClassCastException e) {
7             System.out.println("A class conversion exception occurred:" + e);
8         }
9     }
10 }

```

Run: Main (2) ×

C:\Users\86138\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=57719:E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin" -Didea.config.path=E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\config -Didea.system.path=E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar -Didea.version=2022.3.2 -jar E:\Program Files\JetBrains\IntelliJ IDEA 2022.3.2\bin\idea_rt.jar 57719

A class conversion exception occurred:java.lang.ClassCastException: class java.lang.String cannot be cast to class java.lang.Integer (java.lang.String and java.lang.Integer are not assignable)

Process finished with exit code 0


```
        System.out.print(Color.GREEN.type + " ");  
        break;  
    case BLUE:  
        System.out.print(Color.BLUE.type + " ");  
        break;  
    }  
}  
}
```

实验结果：

2 2 1 2 3 3 1 3 1

进程已结束，退出代码为 0



五、总结

本次实验中我熟悉了继承、多态、接口、抽象类、异常处理以及枚举等相关知识。通过实践了解 java 语言与之前课程所学的 C 和 C++ 语言的区别（比如在继承方面）、重写规则、使用 implements 关键字实现接口、抽象类相关、异常处理和枚举等相关知识。