

## 华东师范大学数据科学与工程学院上机实践报告

课程名称：计算机网络与编程

年级：2022 级

上机实践成绩：

指导教师：张召

姓名：李芳

学号：10214602404

上机实践名称：Java 编程基本语法和基础 2

上机实践日期：2024.03.15

上机实践编号：03

组号：

上机实践时间：

### 一、题目要求及完成情况

#### Task1

- task1: 设计一个名为 Stopwatch (秒表) 的类, 该类继承 Watch (表) 类:

- Watch 类包含: 私有数据域 startTime 和 endTime, 并包含对应公开访问方法: 一个名为 start() 的方法, 将 startTime 重设为当前时间; 一个名为 stop() 的方法, 将 endTime 设置为当前时间;
- Stopwatch 类另包含: 一个名为 getElapsedTime() 的方法, 以毫秒为单位返回秒表记录的流逝时间;
- 在 Main 函数测试 Stopwatch 类功能。

#### 完成 task1 需要的知识:

- 使用 Thread.sleep()方法来实现线程休眠:

**Thread.sleep(long millis):** 使当前线程休眠指定的毫秒数;

**Thread.sleep(long millis, int nanos):** 使当前线程休眠指定的毫秒数和纳秒数。

- java.time.Instant 表示时间戳的类:

1) 获取当前时间的 Instant 对象: **Instant now = Instant.now();**

2) 将 Instant 对象转换为从纪元 (1970 年 1 月 1 日午夜 UTC) 开始的毫秒数:

**long epochMillis = instant.toEpochMilli();**

3) 通过秒数和纳秒数创建 Instant 对象:

**Instant instant = Instant.ofEpochSecond(1615984134, 500000000);**

4) 获取秒数和纳秒数: **long epochSeconds = instant.getEpochSecond();**

**int nano = instant.getNano();**

5) 比较两个 Instant 对象: **int comparison = instant1.compareTo(instant2);**

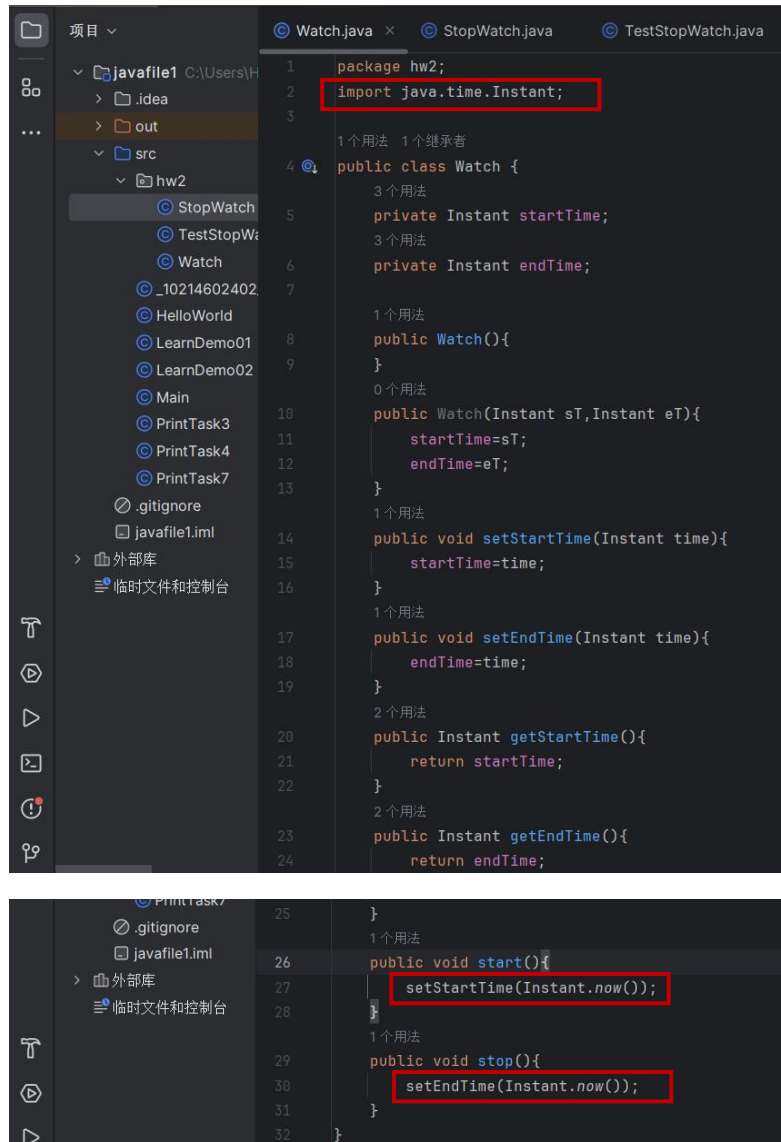
6) 将 Instant 对象转换为 Date 对象: **Date date = Date.from(instant);**

- try{}catch(InterruptedException e){}:**

捕获可能抛出的 InterruptedException 异常 (InterruptedException 是在线程等

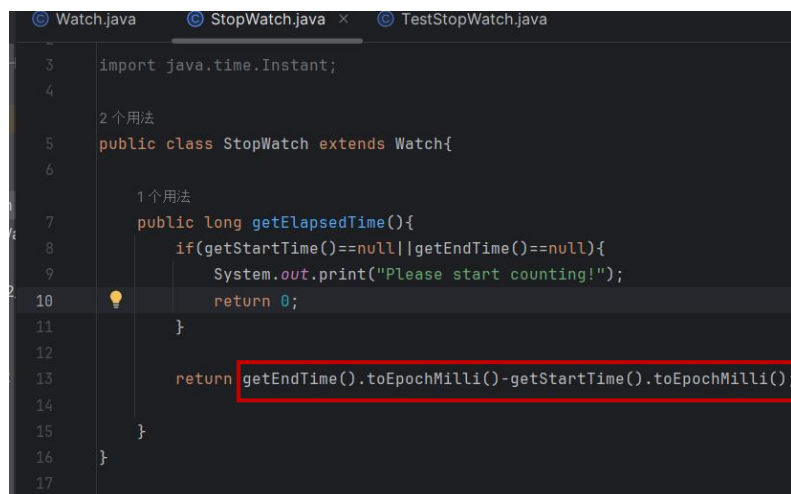
待、睡眠或被中断时可能抛出的异常)

### Watch class:



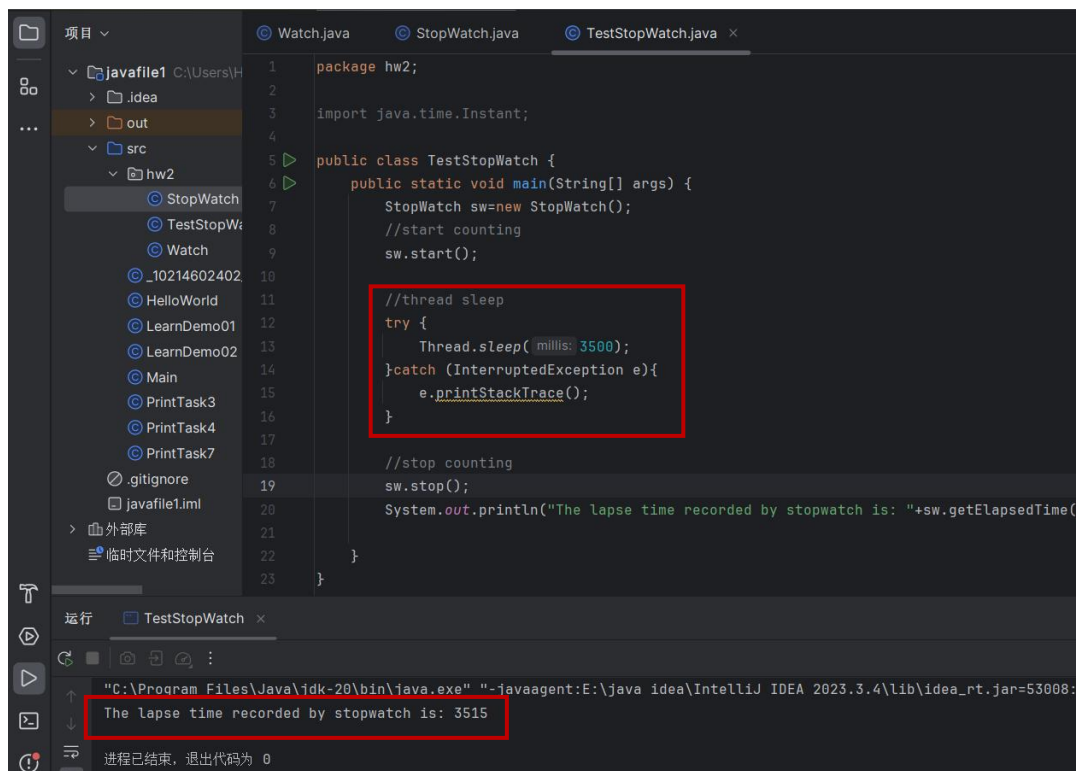
```
1 package hw2;
2 import java.time.Instant;
3
4 public class Watch {
5     private Instant startTime;
6     private Instant endTime;
7
8     public Watch(){
9     }
10    public Watch(Instant sT,Instant eT){
11        startTime=sT;
12        endTime=eT;
13    }
14    public void setStartTime(Instant time){
15        startTime=time;
16    }
17    public void setEndTime(Instant time){
18        endTime=time;
19    }
20    public Instant getStartTime(){
21        return startTime;
22    }
23    public Instant getEndTime(){
24        return endTime;
25    }
26    public void start(){
27        setStartTime(Instant.now());
28    }
29    public void stop(){
30        setEndTime(Instant.now());
31    }
32 }
```

### StopWatch class:



```
1 import java.time.Instant;
2
3 public class StopWatch extends Watch{
4
5     public long getElapsedTime(){
6         if(getStartTime()==null||getEndTime()==null){
7             System.out.print("Please start counting!");
8             return 0;
9         }
10        return getEndTime().toEpochMilli()-getStartTime().toEpochMilli();
11    }
12 }
13 }
```

## Main&Test result:



The screenshot shows the IntelliJ IDEA IDE with the following details:

- Project Structure:** A project named 'javafile1' is open. The source files are located in 'C:\Users\H...\.idea\out\src\hw2'. The files listed are 'StopWatch', 'TestStopWatch', 'Watch', and several other demo files.
- Source Code:** The file 'TestStopWatch.java' is open. It contains the following code:

```
1 package hw2;
2
3 import java.time.Instant;
4
5 public class TestStopWatch {
6     public static void main(String[] args) {
7         Stopwatch sw=new Stopwatch();
8         //start counting
9         sw.start();
10
11         //thread sleep
12         try {
13             Thread.sleep( millis: 3500);
14         }catch (InterruptedException e){
15             e.printStackTrace();
16         }
17
18         //stop counting
19         sw.stop();
20         System.out.println("The lapse time recorded by stopwatch is: "+sw.getElapsedTime());
21     }
22 }
23
```
- Run Output:** The output window shows the command executed: `"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:E:\java_idea\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=53008:...` and the output: `The lapse time recorded by stopwatch is: 3515`.

## Bonus task1

- **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 只提供一种继承方式 extends，是为了简化语言，降低复杂性，使代码更易理解和维护。但也并不意味着它无法像 c++ 一样实现多重继承，java 有接口这一概念，通过接口可以满足多重继承的需求，一个类可以具备多个不同的行为。

## Task2

- **task2:** 对于提供的 Fish 类，实现 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

完成 Task2 所需知识：

1. java.util.Random 类：生成伪随机数。

首先创建一个 Random 对象：Random random = new Random();然后调用下述方法生成不同类型、不同范围的随机数：

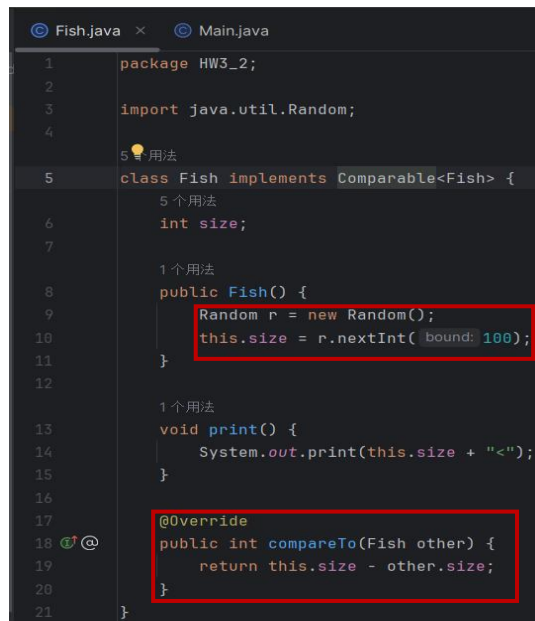
- 1) random.nextInt(), random.nextLong(), random.nextFloat(), random.nextDouble() 等

来生成不同类型的随机整数、长整数、浮点数等。

2) `random.next~(max)`生成`[0,max)`范围随机数, `random.next~(max-min)+min` 生成`[min,max)`范围随机数。

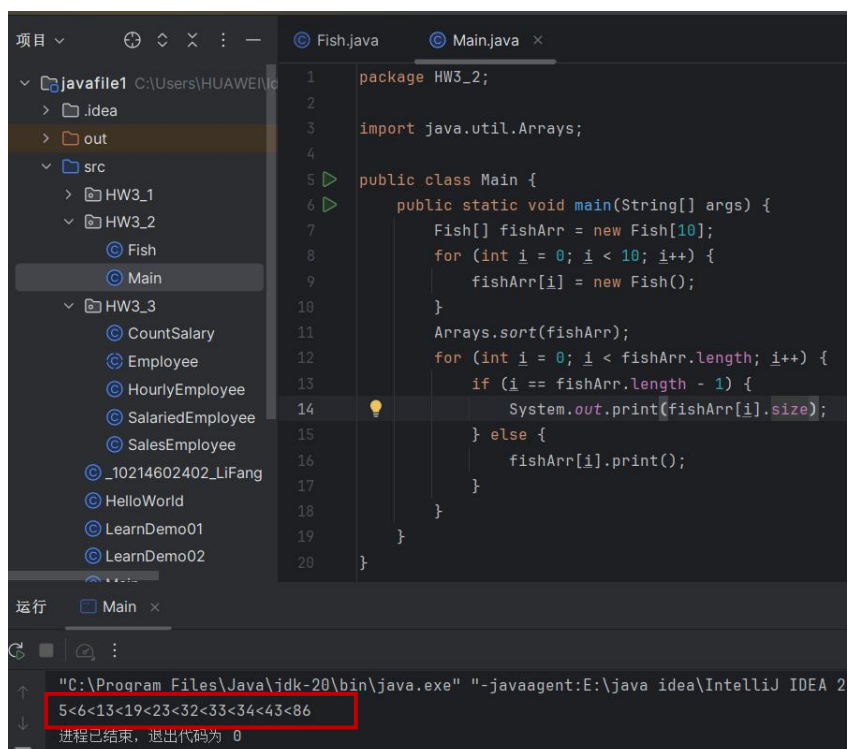
2. 调用 `Comparable` 接口：必须重写覆盖接口中自带的 `compareTo` 方法，再利用 `Arrays.sort()` 进行自动排序。

**Fish class:**



```
1 package HW3_2;
2
3 import java.util.Random;
4
5 class Fish implements Comparable<Fish> {
6     int size;
7
8     public Fish() {
9         Random r = new Random();
10        this.size = r.nextInt( bound: 100);
11    }
12
13    void print() {
14        System.out.print(this.size + "<");
15    }
16
17    @Override
18    public int compareTo(Fish other) {
19        return this.size - other.size;
20    }
21 }
```

**Main&result:**



```
1 package HW3_2;
2
3 import java.util.Arrays;
4
5 public class Main {
6     public static void main(String[] args) {
7         Fish[] fishArr = new Fish[10];
8         for (int i = 0; i < 10; i++) {
9             fishArr[i] = new Fish();
10        }
11        Arrays.sort(fishArr);
12        for (int i = 0; i < fishArr.length; i++) {
13            if (i == fishArr.length - 1) {
14                System.out.print(fishArr[i].size);
15            } else {
16                fishArr[i].print();
17            }
18        }
19    }
20 }
```

运行 Main

"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:E:\java\_idea\IntelliJ IDEA 2021.3.1\lib\idea\_rt.jar=2021.3.1:C:\Program Files\Java\jdk-20\bin" -jar C:\Program Files\Java\jdk-20\bin\java.exe

5<6<13<19<23<32<33<34<43<86

进程已结束，退出代码为 0

### Task3

- **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：

完成 Task3 所需知识：

#### 1. 抽象类：

1) 格式：abstract class 类名字 {}

2) 注意事项：

1. 抽象类不能创建对象，如果创建，编译无法通过而报错。只能创建其非抽象子类的对象。

2. 抽象类中，可以有构造方法，是供子类创建对象时，初始化父类成员使用的。

3. 抽象类中，不一定包含抽象方法，但是有抽象方法的类必定是抽象类。

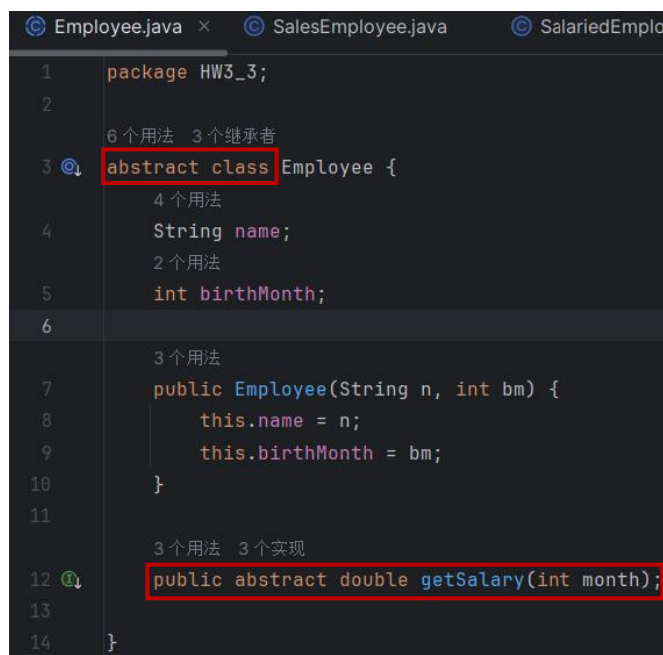
4. 抽象类的子类，必须重写抽象父类中所有的抽象方法，否则，编译无法通过而报错。除非该子类也是抽象类。

#### 2. 抽象方法：

(1)格式：修饰符 abstract 返回值类型 方法名 (参数列表);

(2)注意事项：只包含一个方法名，没有方法体。

**Employee:**



```
1 package HW3_3;
2
3 abstract class Employee {
4     String name;
5     int birthMonth;
6
7     public Employee(String n, int bm) {
8         this.name = n;
9         this.birthMonth = bm;
10    }
11
12    public abstract double getSalary(int month);
13
14 }
```

### SalesEmployee:

```
Employee.java SalesEmployee.java SalariedEmployee.java HourlyEmployee.java
1 package HW3_3;
2
3 class SalesEmployee extends Employee {
4     double saleAmount;
5     double rate;
6     double basicSalary;
7
8     public SalesEmployee(String n, int bm, double sa, double r, double bs) {
9         super(n, bm);
10        this.saleAmount = sa;
11        this.rate = r;
12        this.basicSalary = bs;
13    }
14
15    @Override
16    public double getSalary(int month) {
17        return this.basicSalary + this.rate * this.saleAmount;
18    }
19 }
```

### SalariedEmployee:

```
Employee.java SalesEmployee.java SalariedEmployee.java
1 package HW3_3;
2
3 class SalariedEmployee extends Employee {
4     double monthlySalary;
5
6     public SalariedEmployee(String n, int bm, double ms) {
7         super(n, bm);
8         this.monthlySalary = ms;
9     }
10
11    @Override
12    public double getSalary(int month) {
13        if (month == this.birthMonth) {
14            return monthlySalary + 100.0;
15        } else {
16            return monthlySalary;
17        }
18    }
19 }
```

### HourlyEmployee:



```
Employee.java SalesEmployee.java SalariedEmployee.java HourlyEmployee.java x
1 package HW3_3;
2
3 1个用法
4 class HourlyEmployee extends Employee {
5     2个用法
6     double hourlySalary;
7     2个用法
8     double hour;
9
10    1个用法
11    public HourlyEmployee(String n, int bm, double hs, double h) {
12        super(n, bm);
13        this.hourlySalary = hs;
14        this.hour = h;
15    }
16
17    3个用法
18    @Override
19    public double getSalary(int month) {
20        return hourlySalary * hour;
21    }
22 }
```

### CountSalary&result:

```
项目 项目 项目 项目 项目 项目
Employee.java SalesEmployee.java SalariedEmployee.java HourlyEmployee.java CountSalary.java x
1 package HW3_3;
2
3 public class CountSalary {
4     public static void main(String[] args) {
5         int month = 2;
6         Employee salesEmployee = new SalesEmployee(n: "Xiao Hong", bm: 3, sa: 5500.0, r: 0.1, bs: 2000.0);
7         Employee salariedEmployee = new SalariedEmployee(n: "Xiao Ming", bm: 2, ms: 3000.0);
8         Employee hourlyEmployee = new HourlyEmployee(n: "Xiao Gang", bm: 12, hs: 24.0, h: 100.0);
9         System.out.println(salesEmployee.name + "'s salary is:" + salesEmployee.getSalary(month));
10        System.out.println(salariedEmployee.name + "'s salary is:" + salariedEmployee.getSalary(month));
11        System.out.println(hourlyEmployee.name + "'s salary is:" + hourlyEmployee.getSalary(month));
12    }
13 }
14

运行 CountSalary x
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:E:\java\idea\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=64723:E:\java\idea\IntelliJ IDEA
Xiao Hong's salary is:2550.0
Xiao Ming's salary is:3100.0
Xiao Gang's salary is:2400.0
进程已结束, 退出代码为 0
```

## Task4

- task4: 简要理解并说明: 接口和抽象类在使用场景上的区别

抽象类:

方法: 包含抽象方法、具体方法、抽象类构造方法、静态方法、final 方法。

成员变量：可以有成员变量。

继承：一个类只能继承一个抽象类（单继承）。

适用场景：抽象类适合用于希望在基类中实现一些通用的逻辑，同时又希望留下一些方法需要由子类去实现的情况。

**接口：**

方法：包含抽象方法、默认方法、静态方法和私有方法这几种类型的方法。

成员变量：接口不能包含成员变量，但可以包含常量。

继承：一个类可以实现多个接口（多重继承），具备多种行为。

适用场景：接口适合用于定义规范、约定或者公共行为，使得不同的类可以以统一的方式进行交互。

## Task5

### task5: 请在实验报告中列举出 Error 和 Exception 的区别

二者都属于都属于 Java 中的可抛出对象（Throwable），是 throwable 类的子类，区别如下：

#### Error:

严重程度和原因：Error 表示严重的系统级问题，通常是由于系统内部错误或资源耗尽等导致的。

处理和捕获：Error 通常无法通过代码来处理或捕获，一般由 JVM 抛出并终止程序的执行，程序无法恢复或继续运行。出现 Error 意味着系统已经处于不稳定的状态。

常见种类：

1. OutOfMemoryError: 当 JVM 内存不足以支持应用程序执行时抛出。

常见的子类包括：OutOfMemoryError: 当 JVM 中没有足够的内存分配给新对象时抛出；PermGen space / Metaspace Error: 在持续加载类的情况下，永久代或元空间溢出时抛出。

2. StackOverflowError: 当一个应用程序递归调用层次太深，导致栈空间耗尽时抛出；

3. NoClassDefFoundError: 当 JVM 或 ClassLoader 试图加载类，但找不到类的定义时抛出；



4. `AssertionError`: 当断言语句失败时抛出;
5. `InternalError`: 表示虚拟机内部错误, 如虚拟机性能问题或损坏;
6. `LinkageError`: 当类与其引用的类之间的链接出现问题时抛出, 常见子类有:  
`NoClassDefFoundError`、`UnsupportedClassVersionError` 等。

### Exception:

严重程度和原因: `Exception` 表示一般的异常情况, 通常是由于程序逻辑错误、外部输入错误等导致的。

处理和捕获: `Exception` 可以通过代码进行捕获和处理, 可以使用 `try-catch` 块或 `throws` 关键字来处理异常, 不至于崩溃。受检异常必须在代码中进行处理或声明抛出, 而非受检异常则不需要强制处理。

种类:

1. 编译时异常 (Checked Exception): 常见子类有: `IOException`: I/O 异常, 包括文件读写错误等。 `SQLException`: 数据库访问异常。 `ClassNotFoundException`: 找不到指定类的异常。 `ParseException`: 解析过程中的异常。 `FileNotFoundException`: 未找到文件异常等。
2. 运行时异常 (Runtime Exception): 常见子类有: `NullPointerException`: 当应用程序试图在 `null` 对象上调用方法或访问属性时抛出。 `ArrayIndexOutOfBoundsException`: 数组越界异常。 `ArithmeticException`: 算术运算异常, 如除以零。 `ClassCastException`: 类型转换异常。 `IllegalArgumentException`: 非法参数异常。 `IllegalStateException`: 非法状态异常等。

### Task6

**task6:** 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

完成 Task6 所需知识:

1. `throw` 关键字: 用在方法内, 用来抛出一个异常对象, 将这个异常对象传递到调用者处, 并结束当前方法的执行。格式: `throw new 异常类名(参数);`

2. **Objects** 工具类中的 **requireNonNull** 方法：非空判断，检查一个对象是否为 **null**，如果对象为 **null**，则会抛出 **NullPointerException** 异常。如果对象不为 **null**，则返回该对象。格式：**public static <T> T requireNonNull(T obj){}** or **Objects.requireNonNull(参数, "打印信息");**

3. **throws** 关键字：用于方法声明之上,用于表示当前方法不处理异常,而是提醒该方法的调用者来处理异常(抛出异常)。格式：**修饰符 返回值类型 方法名(参数) throws 异常类名 1,异常类名 2...{ }**

4. **try-catch** 语句块：捕获异常，对异常有针对性的语句进行捕获，可以对出现的异常进行指定方式的处理。**try**：该代码块中编写可能产生异常的代码；**catch**：用来进行某种异常的捕获，实现对捕获到的异常进行处理（**try** 和 **catch** 都不能单独使用,必须连用）。格式：

```
try{  
    编写可能会出现异常的代码  
}catch(异常类型 e){  
    处理异常的代码  
    //记录日志/打印异常信息/继续抛出异常  
}
```

5. 获取异常信息：**Throwable** 类中定义了一些查看方法：

- 1) **public String getMessage()** :获取异常的描述信息,原因(提示给用户的时候,就提示错误原因)。
- 2) **public String toString()** :获取异常的类型和异常描述信息(不用)。
- 3) **public void printStackTrace()** :打印异常的跟踪栈信息并输出到控制台。

6. **finally** 代码块：用于定义在 **try-catch** 结构中无论是否发生异常都会执行的代码块，确保资源得到正确释放或清理，无论是否发生异常。当只有在 **try** 或者 **catch** 中调用退出 **JVM** 的相关方法,此时 **finally** 才不会执行,否则 **finally** 永远会执行。

7. 一次捕获多次处理：一个 **try** 块多个 **catch** 块，多个 **catch** 中的异常不能相同，并且若 **catch** 中的多个异常之间有子父类异常的关系，那么子类异常要求在上面的 **catch** 处理，父类异常在下面的 **catch** 处理。

8. 自定义异常：在开发中根据自己业务的异常情况来定义异常类。做法：

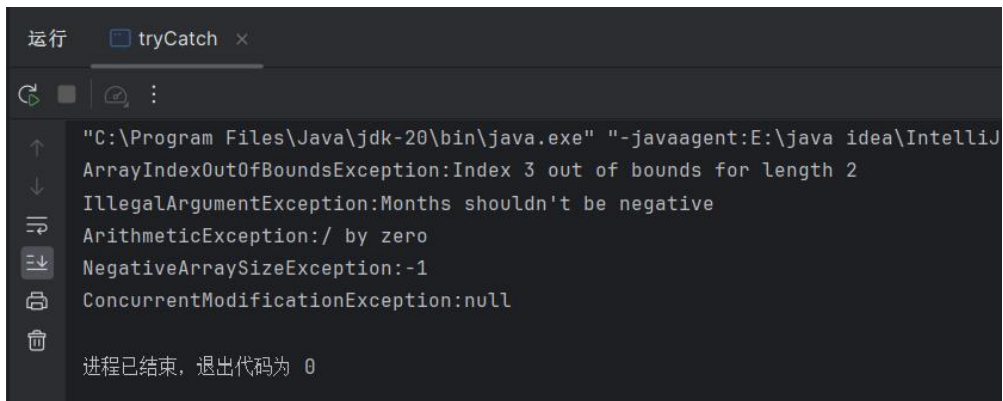
1) 自定义一个编译期异常：自定义类 并继承于 `java.lang.Exception` 。

2) 自定义一个运行时期的异常类：自定义类 并继承于 `java.lang.RuntimeException` 。

### Trycatch:

```
tryCatch.java ×
1 package HW3_6;
2
3 import java.util.ArrayList;
4 import java.util.ConcurrentModificationException;
5 import java.util.List;
6
7 public class tryCatch {
8     public static void main(String[] args) {
9         //1.
10         int[] arr1 = new int[2];
11         try {
12             System.out.println(arr1[3]);
13         } catch (ArrayIndexOutOfBoundsException e1) {
14             System.out.println("ArrayIndexOutOfBoundsException:" + e1.getMessage());
15         }
16
17         //2.
18         int month = -10;
19         try {
20             if (month < 0) {
21                 throw new IllegalArgumentException("Months shouldn't be negative");
22             }
23         } catch (IllegalArgumentException e2) {
24             System.out.println("IllegalArgumentException:" + e2.getMessage());
25         }
26
27         //3.
28         try {
29             int res = 1 / 0;
30             System.out.println(res);
31         } catch (ArithmeticException e3) {
32             System.out.println("ArithmeticException:" + e3.getMessage());
33         }
34
35         //4.
36         try {
37             int[] arr2 = new int[-1];
38         } catch (NegativeArraySizeException e4) {
39             System.out.println("NegativeArraySizeException:" + e4.getMessage());
40         }
41
42         //5.
43         List<Integer> l1=new ArrayList<>(List.of(1,2,3));
44         try {
45             for(Integer i:l1){
46                 l1.remove(i);
47             }
48         } catch (ConcurrentModificationException e5) {
49             System.out.println("ConcurrentModificationException:" + e5.getMessage());
50         }
51     }
52 }
```

## Result:



```
运行 tryCatch x
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:E:\java_idea\IntelliJ
ArrayIndexOutOfBoundsException:Index 3 out of bounds for length 2
IllegalArgumentException:Months shouldn't be negative
ArithmeticException:/ by zero
NegativeArraySizeException:-1
ConcurrentModificationException:null
进程已结束，退出代码为 0
```

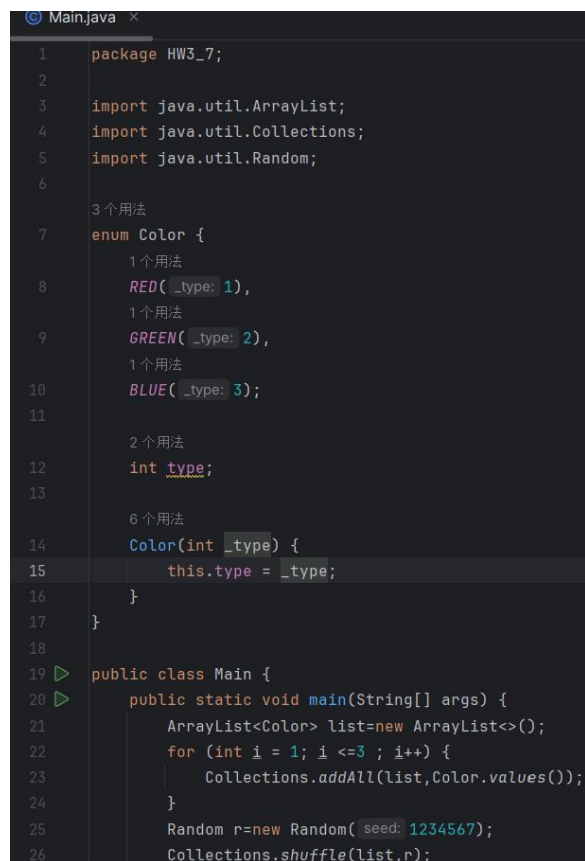
## Task7

- **task7:** 对于 list 中的每个 Color 枚举类，输出其 type（不用换行），请使用 switch 语句实现，请将代码和运行截图附在实验报告中

完成 Task7 所需知识:

1. 枚举类比较特殊，可以在任何方法中（包括主函数）使用枚举类及其枚举常量。
2. Switch 语句中，处理方式相同的 case 可以合并，用，隔开。

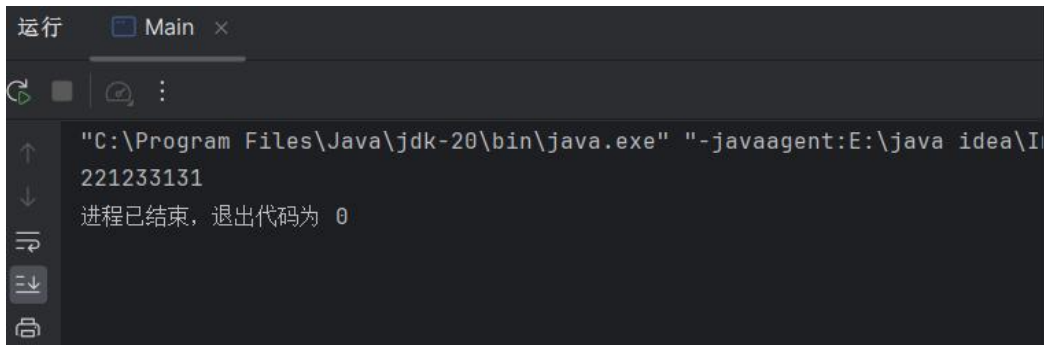
代码:



```
Main.java x
1 package HW3_7;
2
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.Random;
6
7 enum Color {
8     RED(_type: 1),
9     GREEN(_type: 2),
10    BLUE(_type: 3);
11
12    int _type;
13
14    Color(int _type) {
15        this._type = _type;
16    }
17 }
18
19 public class Main {
20     public static void main(String[] args) {
21         ArrayList<Color> list=new ArrayList<>();
22         for (int i = 1; i <=3 ; i++) {
23             Collections.addAll(list,Color.values());
24         }
25         Random r=new Random(_seed: 1234567);
26         Collections.shuffle(list,r);
```

```
27     for (Color c : list) {  
28         switch (c) {  
29             case RED, BLUE, GREEN:  
30                 System.out.print(c.type);  
31                 break;  
32             default:  
33                 throw new RuntimeException("Wrong!");  
34         }  
35     }  
36 }  
37 }
```

**Result:**



```
运行 Main x  
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:E:\java_idea\I  
221233131  
进程已结束，退出代码为 0
```

## Task8

- **task8:** 使用 switch 表达式实现判断某月有多少天的功能（需要考虑闰年），请将代码和运行截图附在实验报告中。

完成 Task8 所需知识:

1. Lambda 表达式的标准格式: (参数类型 参数名称) -> { 代码语句 }

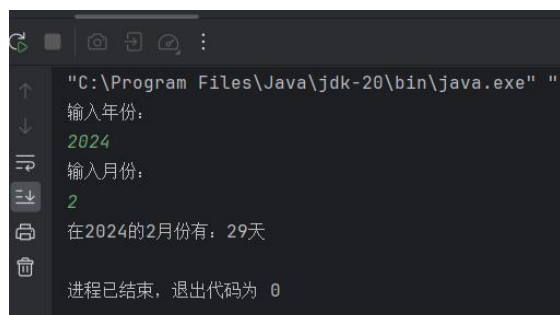
1) 小括号内的语法与传统方法参数列表一致: 无参数则留空; 多个参数则用逗号分隔。

2) -> 是新引入的语法格式, 代表指向动作。

3) 大括号内的语法与传统方法体要求基本一致。

2. yield 关键字: 可以用于从一个 switch 表达式中返回一个值, 类似于 return 关键字, 但主要用于更复杂的情况。

**Result:**



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "  
输入年份:  
2024  
输入月份:  
2  
在2024的2月份有：29天  
进程已结束，退出代码为 0
```

代码:

```
HW3_8(Main.java x)
1 package HW3_8;
2
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7         int month, year;
8         Scanner sc = new Scanner(System.in);
9
10        System.out.println("输入年份: ");
11        year = sc.nextInt();
12
13        System.out.println("输入月份: ");
14        month = sc.nextInt();
15
16        int daysOfMonth=switch (month){
17            case 1,3,5,7,8,10,12 ->31;
18            case 4,6,9,11->30;
19            case 2->{
20                if((year%4==0&&year%100!=0)|| (year%400==0)){
21                    yield 29;
22                }else{
23                    yield 28;
24                }
25            }
26            default -> throw new RuntimeException("输入月份错误!");
27        };
28        System.out.println("在"+year+"的"+month+"月份有: "+daysOfMonth+"天");
29        sc.close();
30    }
31 }
```

## 二、总结

通过本次实验，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。通过每个 task 又加深了对这些知识的理解与巩固：

- 1) 在面向对象编程中，**继承**是一种重要的机制，可以实现代码的复用和扩展。通过创建子类来继承父类的属性和方法，我们可以建立类之间的层级关系，提高代码的灵活性和可维护性。在 Task1 中，通过编写不同层次的类来演示继承的用法，理解了父类和子类之间的关系。
- 2) **多态性**是面向对象编程的一个重要特性，允许不同对象对同一消息做出不同的响应。通过接口和抽象类可以实现多态性，使得程序更加灵活和可扩展。在本次实验的多个 task 中，用到了重写的概念。
- 3) 接口是定义了一组抽象方法的引用类型，可以实现多继承的效果，提供了一种规范化的方式来定义类的行为。通过接口，可以实现类与类之间的解耦，增强代码的灵活性和可



扩展性。在 task2 中，还调用了 Comparable 接口，熟悉了一种自带的排序方式。

- 4) 抽象类是不能被实例化的类，其中可以包含抽象方法和实现方法。抽象类通常用于定义一些通用的行为，子类可以根据需要来实现和扩展。在 task3 中，着重练习了抽象类的编写，子类的实现，深刻地体会了抽象类的作用。在 task4 中，区分了接口和抽象类不同的使用场合。
- 5) 异常处理是 Java 中处理程序运行时错误的重要机制，可以保证程序在发生异常时能够正常运行。通过 try-catch-finally 块，捕获和处理异常，保证程序的稳定性。在 task5 中，区别分清楚了 Exception 和 Error 的区别。在 task6 中，编写了可能会抛出异常的代码，并学会演示如何捕获和处理这些 RuntimeException 异常。
- 6) 枚举是 Java 里一种比较特殊的数据类型，用于定义一组常量。枚举类型可以提高代码的可读性和可维护性。在 task7 中，定义枚举 Color，并学会了如何使用枚举常量。
- 7) Switch 语句是一种语句选择结构，根据不同的条件执行相应的代码块。基本用法跟 C 语言差不多，但是 Java 提供了一些更有趣的用法，比如和 Lambda 共同使用，简化代码。Lambda 表达式是 Java 8 引入的新特性，可以简洁地表示匿名函数。Lambda 表达式可以帮助简化代码，使代码更具可读性和简洁性。在 task8 中，共同使用了 switch 语句和 lambda 表达式来根据不同的条件执行不同的逻辑，还用较简洁的代码完成了实现。