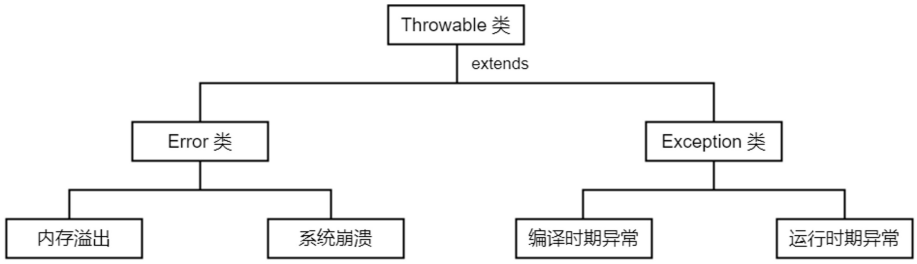
day05_异常、线程

一.异常

异常指的是程序在执行过程中出现的非正常的情况，导致JVM的非正常停止。在Java等面向对象的编程语言中，异常本身就是一个类，产生异常就是创建异常对象并抛出了一个异常对象。

1.异常的体系及分类

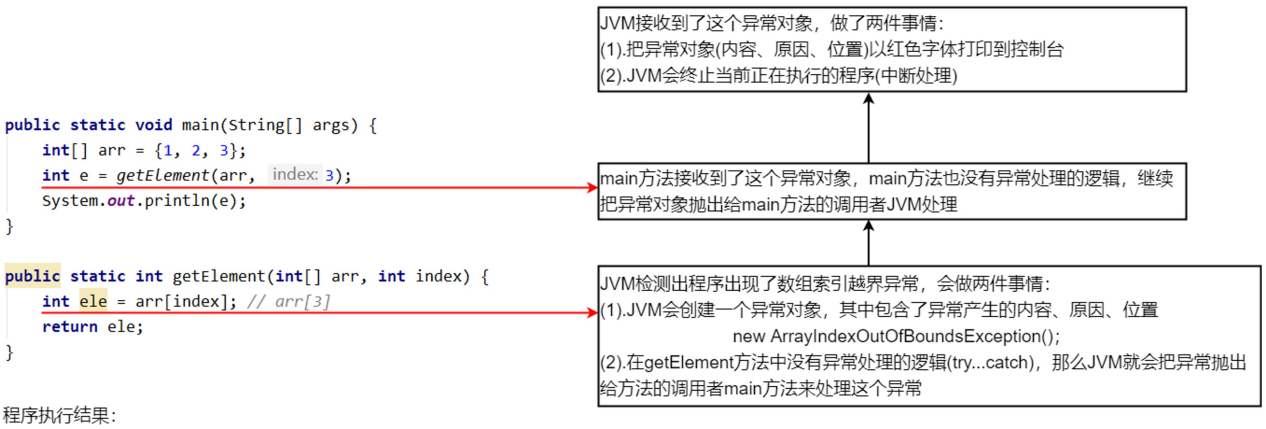
异常的根类是 java.lang.Throwable，其下有两个子类：java.lang.Error 和 java.lang.Exception，平常所说的异常是指 java.lang.Exception 类及其子类的对象。



Error：严重错误，无法通过处理的错误，只能事先避免。

- 编译时期异常：checkedException，在编译时期检查是否有异常。若有异常则编译错误，所以编译时期异常必须进行异常处理。(要么throws，要么try...catch)
- 运行时期异常：runtimeException，在运行时期检查是否有异常。若有异常编译不会报错，运行时会报出异常，所以运行时期异常可以暂不处理，最终会交给JVM处理。

2.异常的产生过程分析




```
public static void main(String[] args) {
    int[] arr = {1, 2, 3};
    int e = getElement(arr, index: 3);
    System.out.println(e);
}

public static int getElement(int[] arr, int index) {
    int ele = arr[index]; // arr[3]
    return ele;
}
```

程序执行结果：

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
    at DemoMain.getElement(DemoMain.java:9)
    at DemoMain.main(DemoMain.java:4)
```

3.异常的处理

(1).使用 throw 关键字抛出异常

(2).Objects 非空判断

(3).使用 throws 关键字处理异常——异常处理的第一种方式

(4).使用 try...catch 处理异常——异常处理的第二种方式

(5).finally 代码块

#.注意事项：

(1).使用 try...catch 捕获多个异常的三种方式

a.多个异常分别处理

```
public static void main(String[] args) {
    try {
        int[] arr = {1, 2, 3};
        System.out.println(arr[5]);
    } catch (ArrayIndexOutOfBoundsException e) {
        e.printStackTrace();
    }

    try {
        List<Integer> list = List.of(1, 2, 3);
        System.out.println(list.get(5));
    } catch (IndexOutOfBoundsException e) {
        e.printStackTrace();
    }
}
```

b.多个异常一次捕获，多次处理：一个try里有多个catch时，catch里边定义的异常变量，如果有父子类关系，那么子类的异常变量必须写在上边，否则就会报错

```
public static void main(String[] args) {
    // ArrayIndexOutOfBoundsException extends IndexOutOfBoundsException
    try {
        int[] arr = {1, 2, 3};
        System.out.println(arr[5]);
        List<Integer> list = List.of(1, 2, 3);
        System.out.println(list.get(5));
    } catch (ArrayIndexOutOfBoundsException e) {
        e.printStackTrace();
    } catch (IndexOutOfBoundsException e) {
        e.printStackTrace();
    }
}
```

c.多个异常一次捕获一次处理：使用父类异常来接收

```java
public static void main(String[] args) {
    try {
        int[] arr = {1, 2, 3};
        System.out.println(arr[5]);
        List<Integer> list = List.of(1, 2, 3);
        System.out.println(list.get(5));
    } catch (Exception e) { // 或者 IndexOutOfBoundsException e
        e.printStackTrace();
    }
}
```

(2).finally 中含有 return 语句：如果 finally 有 return 语句，永远返回 finally 中的结果，应该避免在 finally 中使用 return

```java
public class DemoMain {
    public static void main(String[] args) {
        System.out.println(getA()); // 100
    }

    // 定义一个方法返回变量a的值
    private static int getA() {
        try {
            int a = 10;
            return a;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            int a = 100;
            return a;
        }
    }
}
```

(3).父子类的异常

- 如果父类方法抛出了多个异常，那么子类重写该父类方法时，必须抛出和父类相同的异常、或父类异常的子类、或不抛出异常。
- 如果父类方法没有抛出异常，那么子类重写父类该方法时也不可抛出异常。如果该子类重写的方法产生了异常，只能捕获处理，不能声明抛出。

二.自定义异常

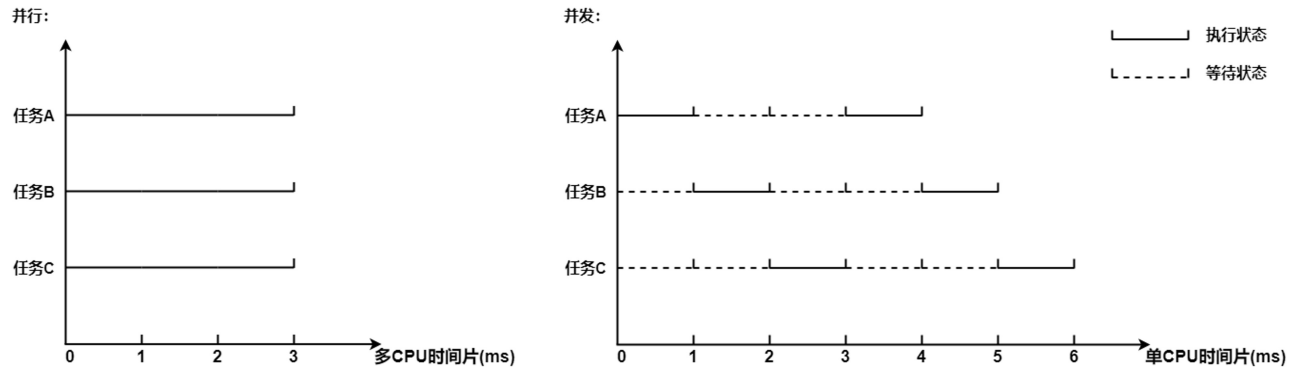
三.线程

1.并行与并发

- 并行：指两个或多个任务在同一时刻发生（同时发生）
- 并发：指两个或多个任务在同一个时间段内执行

在单 CPU 的系统中，每一时刻只能有一个程序在运行，所以是实现不了程序并行运行的。但是可以在一段时间内让程序轮流执行，即

并发执行。由于 CPU 在各个程序之间的切换速度非常之快，导致在宏观上给人的感觉是同时运行的，但是在微观上的一个时间段这些程序还是交替轮流执行的。而在多个 CPU 系统中，这些程序可以被分配到多个 CPU 上，实现多程序并行执行。



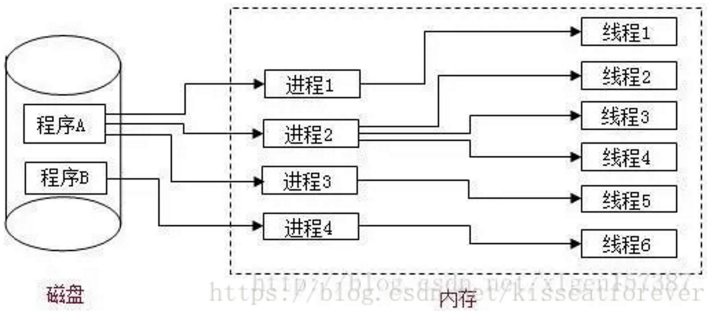
2.进程和线程

当我们在内存中运行一个程序时，操作系统就要为这个 "正在内存中运行的程序" 分配资源(硬件资源、软件资源)用来支持这个程序的运行。操作系统在给各个 "正在内存中运行的程序" 分配系统资源的时候，是以进程为单位进行分配的。

例如：运行QQ程序时，操作系统就为其开了一个进程；运行迅雷程序时，操作系统就又为其开了一个进程。所以运行某个软件、程序时，操作系统就会为其开一个进程用来给其分配系统资源。在QQ的这个进程里，传输文字开一个线程，传输语音开一个线程，弹出对话框又开了一个线程。在软件运行的过程中（在这个进程里面），多个子任务分别支撑完成QQ的运行，那么这些子任务分别有一个线程。

- 进程(Process)：是计算机中的程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位。即进程是程序在内存中的一次执行过程(动态的概念)，是系统运行程序的基本单位。一个程序运行后至少有一个进程，也可以同时拥有多个进程，运行一个程序序即是一个或多个进程从创建、运行到消亡的过程。

- 线程(Thread)：是进程中的一个执行单元，是比进程更小的能独立运行的基本单位。一个进程中至少有一个线程，一个进程中是可以有多个线程的，这个应用程序被称为多线程程序。



#.线程的调度：单 CPU 系统是不能并行处理多个任务的，只能是多个任务在单 CPU 上并发运行。线程也是一样，从宏观上看线程是并行运行的，但从微观上分析却是串行运行的，即一个线程一个线程的交替轮流运行，所以就要对多个线程的执行顺序进行调度。

a.分时调度：所有线程轮流使用 CPU ，并且平均分配每个线程占用 CPU 的时间

b.抢占式调度：优先让优先级高的线程使用 CPU，如果线程的优先级相同，会随机选择一个(线程随机性)，Java 使用的是抢占式调度

3.Java 主线程

当我们使用 java 命令执行一个类的时候，都会启动一个 java 虚拟机，操作系统就会为 JVM 启动一个进程。当 JVM 去执行 main 方法时，main 方法会进入到栈内存中。此时 JVM 会找操作系统开辟一个线程用来执行main 方法，这个用来执行 main 方法的线程就叫主(main)线程。

4.创建多线程程序的第一种方式——定义一个线程类继承 Thread 类