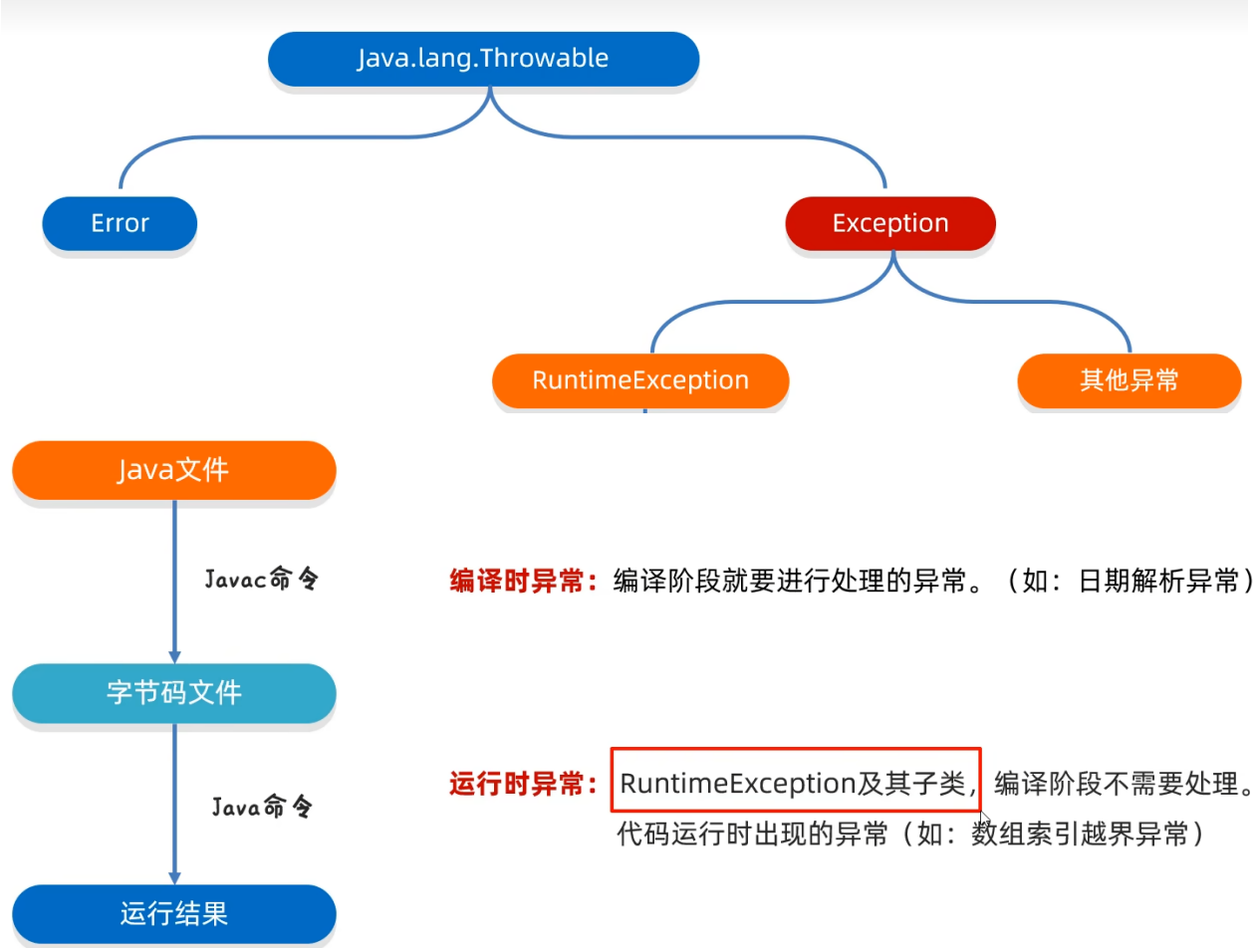
23 异常体系介绍

- 1 异常

  异常：异常就是代表程序出现的问题

- 2 分类



  - Error

    Error：代表的系统级别错误（属于严重问题）

    系统一旦出现问题，sun公司会把这些错误封装成Error对象。

    Error是给sun公司自己用的，不是给我们程序员用的。

    因此我们开发人员不用管它。

  - Exception(异常)

    Exception：叫做异常，代表程序可能出现的问题。

    我们通常会用Exception以及他的子类来封装程序出现的问题。

    运行时异常： RuntimeException及其子类，编译阶段不会出现异常提醒。

    运行时出现的异常（如：数组索引越界异常）

    编译时异常：编译阶段就会出现异常提醒的。（如：日期解析异常）

- 3 代码实现

```java
//编译时异常(在编译阶段，必须要手动处理，否则代码报错)
String time = "2030年1月1日";
SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy年MM月dd日");
Date date = sdf.parse(time);
System.out.println(date);

//运行时异常
int[] arr = {1,2,3,4,5};
System.out.println(arr[10]);//ArrayIndexOutOfBoundsException
```

- 4 作用

作用一： 异常是用来查询bug的关键参考信息

作用二： 异常可以作为方法内部的一种特殊返回值，以便通知调用者底层的执行情况

- 异常从下往上看

```java
public void setAge(int age) {
    if(age < 18 || age > 40){
        //System.out.println("年龄超出范围");
        throw new RuntimeException();
    }else{
        this.age = age;
    }
}
```

- 5 处理方式
  - 5.1 JVM 默认的处理方式
    - 把异常的名称，异常原因及异常出现的位置等信息输出在了控制台
    - 程序停止执行，下面的代码不会再执行了
  - 5.2 自己处理（捕获异常）

格式：

```java
try {
    可能出现异常的代码;
} catch(异常类名 变量名) {
    异常的处理代码;
}
```

目的： 当代码出现异常时，可以让程序继续往下执行。

```
try{
    //可能出现异常的代码;
    System.out.println(arr[10]);
}catch(ArrayIndexOutOfBoundsException e){
    //如果出现了ArrayIndexOutOfBoundsException异常，我该如何处理
    System.out.println("索引越界了");
}
```

- **ctrl+alt+T快捷键**
- **问1：如果try中没有遇到问题，怎么执行?**
  - 会把try里面的代码全部执行完毕，不会执行catch里面的代码。只有出现异常才会执行catch里面的代码
- **问2：如果try中遇到多个问题，怎么执行?**
  - 会写多个catch与之对应。如果我们要捕获多个异常，这些异常中如果存在**父子关系**，那么**父类**一定要写在**下面**。
  - 在JDK7以后，我们可以在catch中同时捕获多个异常，中间用 | 进行隔开。表示出现了A/B异常的话，采取同一种处理方案。

    ```
    catch(ArrayIndexOutOfBoundsException | ArithmeticException e){
       System.out.println("索引越界了");
    ```
- **问3：如果try中遇到的问题没有被捕获，怎么执行?**
  - 相当于try.....catch的代码白写了，最终还是会交给虚拟机进行处理。
- **问4：如果try中遇到了问题，那么try下面的其他代码还会执行吗?**
  - 下面的代码不会执行了，直接跳转到对应的catch当中，执行catch里面的语句体。但是如果没有对应的catch与之匹配，那么还是会交给虚拟机进行处理。

- **5.3 Throwable的成员方法**

方法名称	说明
public String getMessage()	返回此 throwable 的详细消息字符串
public String toString()	返回此可抛出的简短描述
public void printStackTrace()	把异常的错误信息输出在控制台

```
try {
    System.out.println(arr[10]);
} catch (ArrayIndexOutOfBoundsException e) {
  /*  String message = e.getMessage();
    System.out.println(message);//Index 10 out of bounds for length 6*/

 /*   String str = e.toString();
    System.out.println(str);//java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 6*/

    e.printStackTrace();
```

- **printStackTrace 仅仅是打印信息，不会停止程序运行。**

```java
//正常的输出语句
System.out.println(123);

//错误的输出语句（而是用来打印错误信息）
System.err.println(123);
```

- **5.4 抛出处理**

throws

注意： 写在方法定义处，表示声明一个异常
告诉调用者，使用本方法可能会有哪些异常

```java
public void 方法()throws 异常类名1,异常类名2...{
        ...
}
```

throw

注意： 写在方法内，结束方法
手动抛出异常对象，交给调用者
方法中下面的代码不再执行了

```java
public void 方法(){
     throw new NullPointerException();
}
```

- 编译时异常：必须要写。
- 运行时异常：可以不写。

```java
public static int getMax(int[] arr) throws NullPointerException,ArrayIndexOutOfBoundsException{
    if(arr == null){
        //手动创建一个异常对象，并把这个异常交给方法的调用者处理
        //此时方法就会结束，下面的代码不会再执行了
        throw new NullPointerException();
    }

    if(arr.length == 0){
        //手动创建一个异常对象，并把这个异常交给方法的调用者处理
        //此时方法就会结束，下面的代码不会再执行了
        throw new ArrayIndexOutOfBoundsException();
    }


    System.out.println("看看我执行了吗？");
    int max = arr[0];
    for (int i = 1; i < arr.length; i++) {
        if(arr[i] > max){
            max = arr[i];
        }
    }
    return max;
}
```

- **总结**

1. 虚拟机默认处理异常的方式

   把异常信息以红色字体打印在控制台，并结束程序

2. 捕获：try...catch

   一般用在调用处，能让代码继续往下运行。

3. 抛出：throw throws

   在方法中，出现异常了。

   方法就没有继续运行下去的意义了，采取抛出处理。

   让该方法结束运行并告诉调用者出现了问题。

- **6 自定义异常**
  - **步骤：1.定义异常类 2.写继承关系 3.空参构造 4.带参构造**

    意义：就是为了让控制台的报错信息更加的见名之意