实验三 一般异常技术、自定义异常技术

任务完成收获：

1对类的继承、多态和异常处理有了更深刻的理解​

1. 熟练掌握了接口的定义与实现​：完成了多个接口的定义，以及类对接口的实现，学会了如何通过接口定义抽象方法、默认方法和静态方法，并在实现类中进行相应的实现和调用，如在“接口的基本定义与实现”和“接口的默认方法与静态方法”代码示例中，分别定义了接口并实现了接口中的方法，加深了对Java接口机制的理解。​
2. 深入理解了 final 修饰符的作用和自定义异常的创建与使用​：通过定义 final 修饰的类、方法和变量，明确了 final 关键字在防止继承、重写和变量重新赋值方面的作用。同时，在“自定义异常（业务逻辑异常）”的代码示例中，创建了自定义异常类，并在方法中抛出和捕获自定义异常，学会了如何根据业务需求定义和使用异常来处理特定的错误情况。
3. 提升了代码的组织和规范性​：在编写代码的过程中，注重了代码的结构和可读性，遵循良好的命名规范代码格式，用注释对代码进行了清晰的说明，提高代码可维护性可理解性。助于团队开发中更好地协作和交流，后续代码维护扩展奠定基础。
4. 5通过实现不同类之间的继承关系，重写方法来实现多态，以及处理各种异常，进一步掌握了面向对象编程的核心概念

​

**实验目的：**

1.掌握Java中一般异常（编译时异常和运行时异常）的处理机制，包括**try-catch-finally**和**throws**的使用。

2.理解自定义异常的应用场景，学会通过继承**Exception**或**RuntimeException**定义符合业务逻辑的异常类。

3.提升程序健壮性，能够合理设计异常处理逻辑解决实际问题。

实验题目：

**二、实验环境：**

**IntelliJ IDEA Community Edition 2024.3.5**

**JDK**

### ****题目 1：编译时异常处理（ClassNotFoundException）****

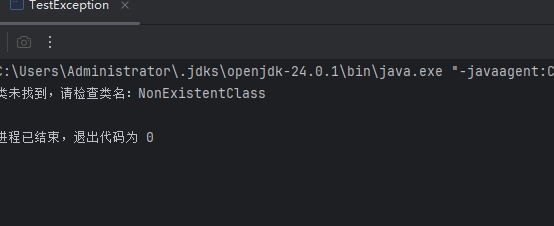
编写一个程序，动态加载一个类（如 **com.example.TestClass**），并输出“类加载成功！”。要求：

使用 **Class.forName()**加载类，该方法会抛出 **ClassNotFoundException**。

用 **try-catch** 捕获该异常，若类不存在，提示用户“类未找到，请检查类名”。

在 **finally**块中输出“加载流程结束”，确保无论是否异常都会执行。



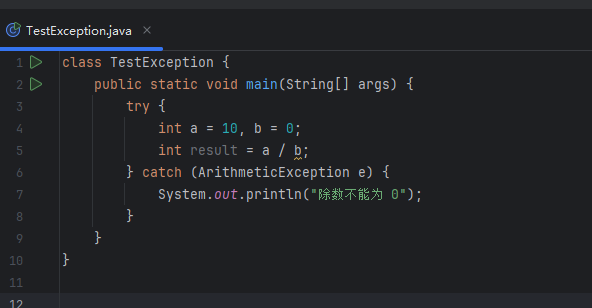


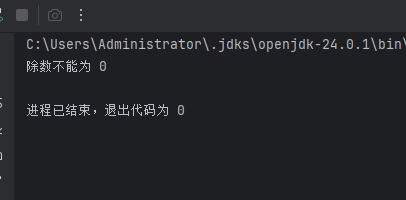
### ****题目 2：运行时异常处理（ArithmeticException）****

编写一个方法 **divide(int a, int b)，**计算 **a / b**的结果。要求：

若 **b = 0**，抛出 **ArithmeticException** 并提示“除数不能为 0”。

在 **main**方法中调用 **divide**，用 **try-catch**捕获异常并处理。





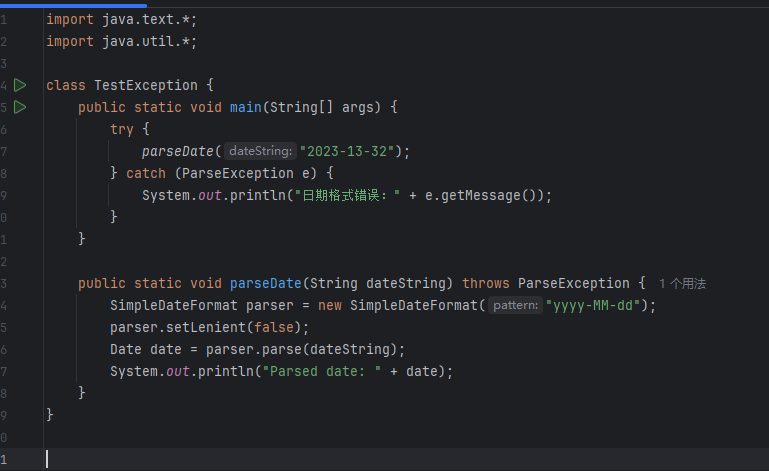
### ****题目 3：throws 声明异常（ParseException - 日期解析异常）****

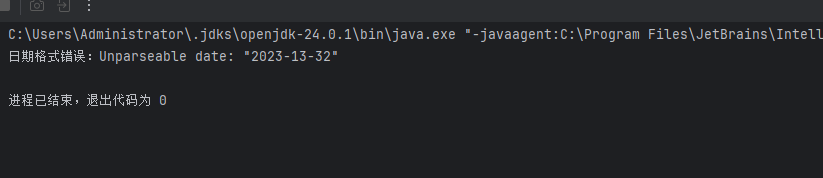
编写一个方法**parseDate(String dateStr)**，将字符串 **dateStr**（如 **"2023-12-31"**）解析为 **Date**对象。要求：

使用 **SimpleDateFormat.parse()**，该方法会抛出 **ParseException**（编译时异常）。

在方法签名中用 **throws**声明该异常，由调用者处理。

在 **main**方法中调用 **parseDate**，并用 **try-catch**捕获异常，提示用户“日期格式错误”。





### ****题目 4：自定义异常（业务逻辑异常）****

定义一个自定义异常 **InvalidAgeException**，表示年龄无效（如年龄 < 0 或 > 120）。  
编写一个方法 **setAge(int age)，**若年龄无效，抛出 **InvalidAgeException。**  
在 **main**方法中调用 **setAge**，捕获并处理该异常。



