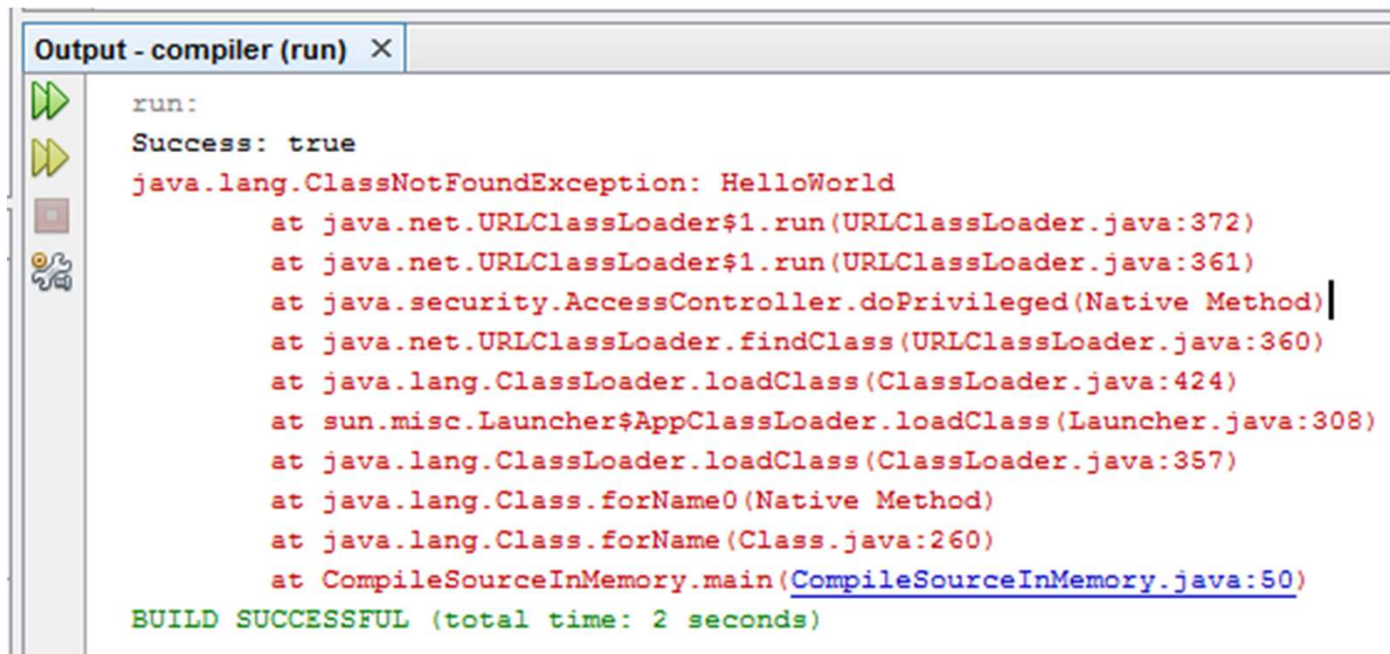


# 第8章 例外处理

# Java 异常[例外]类 (exception)

- 执行Java代码时，可能会发生编译时语法错误或运行时的异常[例外]:
  - 代码少了分号，会抛出java.lang.Error的语法错误
  - 运算除法时分母设为了0，则会抛出 java.lang.ArithmeticException 的异常[例外]

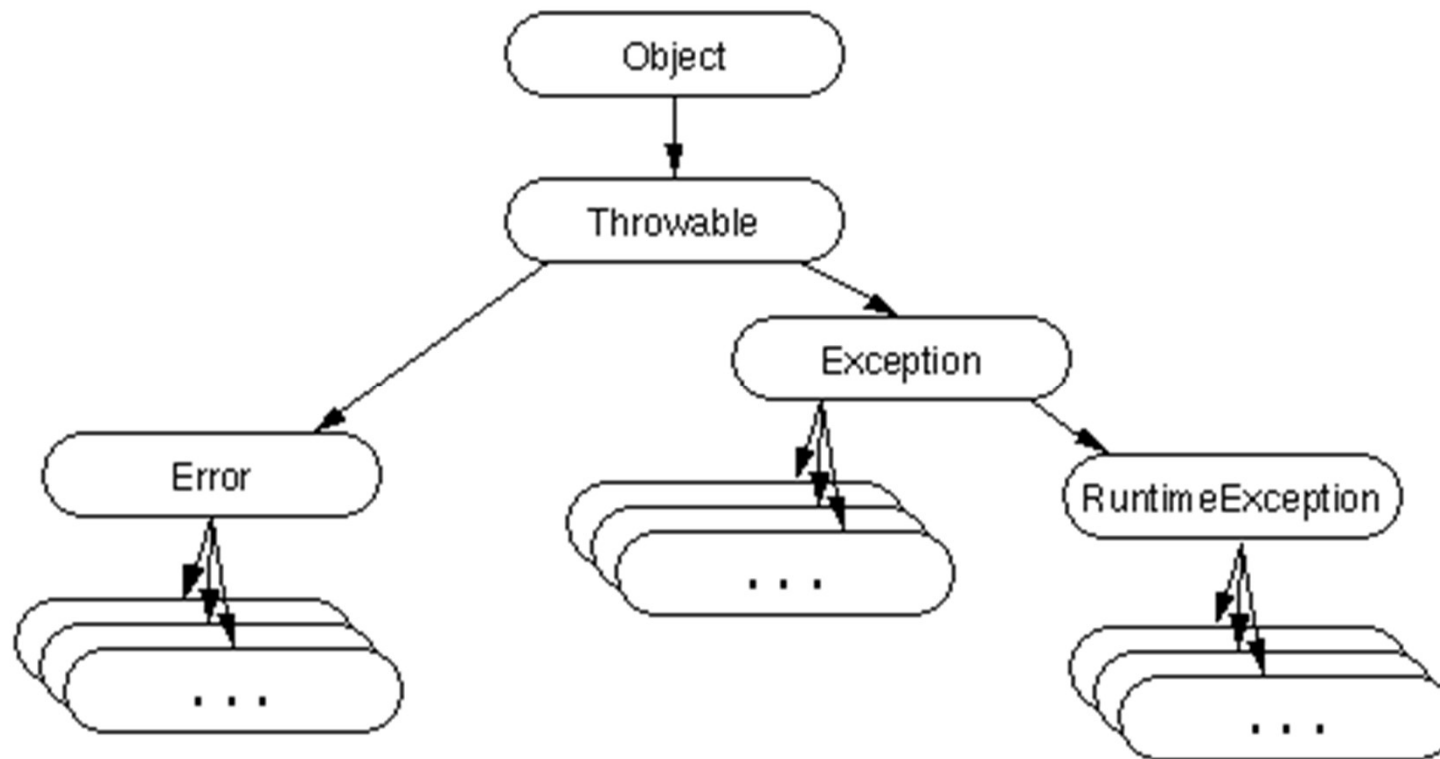


```

Output - compiler (run) x
run:
Success: true
java.lang.ClassNotFoundException: HelloWorld
    at java.net.URLClassLoader$1.run(URLClassLoader.java:372)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:260)
    at CompileSourceInMemory.main(CompileSourceInMemory.java:50)
BUILD SUCCESSFUL (total time: 2 seconds)
    
```

# Java 异常[例外]类 (exception)

- 所有的异常[例外]类都是从 `java.lang.Exception` 类继承的子类。
- 常见的异常类有 [IOException](#) 类和 [RuntimeException](#) 类。





# Java 异常[例外]类 (exception)

- 异常[例外]类的对象的常用方法

```
public String getMessage()
```

返回关于发生的异常的详细信息。这个消息在Throwable 类的构造函数中初始化了。

```
public Throwable getCause()
```

返回一个Throwable 对象代表异常原因。

```
public String toString()
```

使用getMessage()的结果返回类的串级名字。

```
public void printStackTrace()
```

打印toString()结果和栈层次到System.err，即错误输出流。

```
public StackTraceElement [] getStackTrace()
```

返回一个包含堆栈层次的数组。下标为0的元素代表栈顶，最后一个元素代表方法调用堆栈的栈底。

```
public Throwable fillInStackTrace()
```

用当前的调用栈层次填充Throwable 对象栈层次，添加到栈层次任何先前信息中。

## Java 异常[例外]处理 – try...catch...

- 如何侦测异常[例外]的发生，并作出相应的措施？ ---> 捕获异常[例外]
- 使用 try 和 catch 关键字可以捕获异常[例外]。
- try/catch 代码块放在异常[例外]可能发生的地方。
- try/catch 的语法如下：

```
try {  
    // Block of code to try  
}  
catch(Exception e) {  
    // Block of code to handle errors  
}
```

- 在try语句里定义想要侦测异常[例外]的代码块
- 如果发生catch后面括号里指定的异常[例外]e时，则运行catch的代码块。

# Java 异常[例外]处理 – try...catch...

- 考虑以下示例：
  - 将产生一个错误，因为myNumbers [10]不存在。

```
public class MyClass {  
    public static void main(String[ ] args) {  
        int[] myNumbers = {1, 2, 3};  
        System.out.println(myNumbers[10]); // error!  
    }  
}
```

- 输出将是这样的：

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10  
    at MyClass.main(MyClass.java:4)
```

## Java 异常[例外]处理 – try...catch...

- 前面的代码用try...catch...来写的话，就可以捕获异常[例外]
- 尝试代码：TryCatch.java

```
public class MyClass {  
    public static void main(String[] args) {  
        try {  
            int[] myNumbers = {1, 2, 3};  
            System.out.println(myNumbers[10]);  
        } catch (Exception e) {  
            System.out.println("Something went wrong.");  
        }  
    }  
}
```

## Java 异常[例外]处理 – 多重捕获

- 一个 try 代码块后面跟随多个 catch 代码块的情况就叫多重捕获

```
try{  
    // 程序代码  
}catch(异常类型1 异常的变量名1){  
    // 程序代码  
}catch(异常类型2 异常的变量名2){  
    // 程序代码  
}catch(异常类型2 异常的变量名2){  
    // 程序代码  
}
```



## Java 异常[例外]处理 – finally关键字

- 无论是否发生异常[例外]，finally代码块都会被执行
- finally代码块放在最后：

```
try{  
    // 程序代码  
}catch(异常类型1 异常的变量名1){  
    // 程序代码  
}catch(异常类型2 异常的变量名2){  
    // 程序代码  
}finally{  
    // 程序代码  
}
```

# Java 异常[例外]处理 - throw关键字

- 注意是throw不是throw~~s~~
- 可以用throw抛出一个自己创建的异常[例外]对象
- 比如：如果年龄低于18岁，则会引发异常[例外]（输出“拒绝访问”）。如果年龄在18岁以上，输出“已授予访问权限”：

```
public class MyClass {
    static void checkAge(int age) {
        if (age < 18) {
            throw new ArithmeticException("Access denied - You must be at least 18 years old.");
        }
        else {
            System.out.println("Access granted - You are old enough!");
        }
    }

    public static void main(String[] args) {
        checkAge(15); // Set age to 15 (which is below 18...)
    }
}
```

```
Exception in thread "main" java.lang.ArithmeticException: Access denied - You must be at least 18 years old.
    at MyClass.checkAge(MyClass.java:4)
    at MyClass.main(MyClass.java:12)
```

## Java 异常[例外]处理 – throws关键字

- 注意这次是throws
- 放在方法参数的括号后面，表示什么异常[例外]类型可通过此方法抛出

```
import java.io.*;
public class className
{
    public void deposit(double amount) throws RemoteException
    {
        // Method implementation
        throw new RemoteException();
    }
    //Remainder of class definition
}
```

Q&A

You Have  
Questions  
We Have  
Answers



THANK YOU