

# Chp11 异常

## Key Point

- 异常的概念和分类
- 异常的产生和传递
- 异常的处理
- 自定义异常

### 练习

#### 1. 填空

Java 中所有的错误都继承自\_\_\_\_\_类；在该类的子类中，\_\_\_\_\_类表示严重的底层错误，对于这类错误一般处理的方式是\_\_\_\_\_；\_\_\_\_\_类表示例外、异常。

#### 2. 查 api, 填空

异常类 `java.rmi.AlreadyBoundException`，从分类上说，该类属于\_\_\_\_\_（已检查| 未检查）异常，从处理方式上说，对这种异常\_\_\_\_\_；异常类 `java.util.regex.PatternSyntaxException`，从分类上说，该类属于\_\_\_\_\_（已检查|未检查）异常，从处理方式上说，对这种异常\_\_\_\_\_。

#### 3. （异常的产生）把下面代码补充完整

```
public class TestThrow{
    public static void main(String args[]){
        throwException(10);
    }
    public static void throwException(int n){
        if (n == 0){
            //抛出一个 NullPointerException
        }else{
            //抛出一个 ClassCastException
            //并设定详细信息为“类型转换出错”
        }
    }
}
```

#### 4. （try-catch-finally）有如下代码：

```
import java.io.*;
import java.sql.*;
class TestException{
    public static void main(String args[]){
        System.out.println("main 1");
        int n;
        //读入 n
        ma(n);
    }
}
```

```

        System.out.println("main2");
    }
    public static void ma(int n){
        try{
            System.out.println("ma1");
            mb(n);
            System.out.println("ma2");
        }catch(EOFException e){
            System.out.println("Catch EOFException");
        }catch(IOException e){
            System.out.println("Catch IOException");
        }catch(SQLException e){
            System.out.println("Catch SQLException");
        }catch(Exception e){
            System.out.println("Catch Exception");
        }finally{
            System.out.println("In finally");
        }
    }
}
public static void mb(int n) throws Exception{
    System.out.println("mb1");
    if (n == 1) throw new EOFException();
    if (n == 2) throw new FileNotFoundException();
    if (n == 3) throw new SQLException();
    if (n == 4) throw new NullPointerException();
    System.out.println("mb2");
}
}

```

问：当读入的  $n$  分别为 1, 2, 3, 4, 5 时，输出的结果分别是什么？

5. （自定义异常）创建两个自定义异常类 `MyException1` 和 `MyException2`。

要求：

- 1) `MyException1` 为已检查异常，`MyException2` 为未检查异常
- 2) 这两个异常均具有两个构造函数，一个无参，另一个带字符串参数，参数表示产生异常的详细信息。

6. （自定义异常）在上一题的基础上，把下面代码补充完整。

```
public class TestMyException{
    public static void main(String args[]){
        int n;
        //读入 n
        try{
            m(n);
        }catch(MyException1 ex1){
            //输出 ex1 详细的方法调用栈信息
        }catch(MyException2 ex2){
            //输出 ex2 的详细信息
            //并把 ex2 重新抛出
        }
    }
}

public static void m(int n)_____ { //声明抛出 MyException1
    if (n == 1) {
        //抛出 MyException1
        //并设定其详细信息为“n == 1”
    }else {
        //抛出 MyException2
        //并设定其详细信息为“n == 2”
    }
}
}
```

7. （try-catch）代码改错。

```
class MyException{}
class TestException{
    public static void main(String args[]){
        ma();
    }
    public static int ma(){
        try{
            m();
            return 100;
        }catch(Exception e){
            System.out.println("Exception");
        }
        catch(ArithmeticException e){
            System.out.println("ArithmeticException");
        }
    }
}
```

```

    }
    public static void m(){
        throw new MyException();
    }
}

```

8. （方法覆盖）有如下代码

```

import java.io.IOException;
class Super{
    public void ma() throws IOException{}
}
interface IA{
    void mb();
}
public class MySub extends Super implements IA{
    public void ma() //1_____ {
    }
    public void mb() //2_____ {
    }
}

```

问：

在//1 处，填入以下\_\_\_\_\_代码可以编译通过，在//2 处，填入\_\_\_\_\_代码可以编译通过。

- A. throws java.io.IOException
- B. throws java.io.FileNotFoundException, java.io.EOFException
- C. throws java.sql.SQLException
- D. 不能抛出任何异常。

9. \*（异常处理）有如下代码

```

import java.io.*;
import java.sql.*;
public class TestTryCatch{
    public static void main(String args[]){
        try{
            ma(10);
            System.out.println("No Exception");
        }
        catch EOFException ex1 {
            System.out.println("ex1");
        }
        catch IOException ex2 {
            System.out.println("ex2");
        }
        catch(SQLException ex3) {

```

```

        System.out.println("ex3");
    }
}
public static void ma(int n) throws Exception{
    if (n == 1){
        throw new IOException();
    }else if (n == 2){
        throw new EOFException();
    }else if (n == 3) {
        throw new SQLException();
    }
}
}
}

```

选择正确答案：

- A. 编译不通过
- B. 编译通过，输出 No Exception
- C. 编译通过，输出 ex1
- D. 编译通过，输出 ex2
- E. 编译通过，输出 ex3

10. \*（try-catch，局部变量）有如下代码

```

public class TestTryCatch{
    public static void main(String args[]){
        System.out.println( ma() );
    }
    public static int ma(){
        int n;
        try{
            n = 10/0;
        }catch(Exception e){
        }
        return n;
    }
}

```

选择正确答案：

- A. 编译不通过
- B. 编译通过，输出-1
- C. 编译通过，输出 0

11. \*（try-catch-finally）有如下代码

```

public class TestFinally{
    public static void main(String args[]){
        System.out.println ( ma() );
    }
    public static int ma(){

```

```

        int b;
        //读入 b
        try{
            int n = 100;
            return n/b;
        }catch(Exception e){
            return 10;
        }finally{
            return 100;
        }
    }
}

```

在 `ma` 中，当读入的 `b` 为 100 时，输出结果为\_\_\_\_，当读入的 `b` 为 0 时，输出结果为\_\_\_\_\_。

12. \* (try-finally) 写出下面代码运行的结果

```

public class TestTryFinally{
    public static void main(String args[]){
        try{
            ma();
        }catch(Exception ex1){
        }
    }
    public static void ma() throws Exception {
        int n = 10;
        int b;
        //读入一个整数 b
        try{
            System.out.println("ma1");
            int result = n / b;
            System.out.println("ma2 " + result);
        }finally{
            System.out.println("In Finally");
        }
    }
}

```

在 `ma` 中，读入整数 `b`，如果读入的值为 10，则输出：  
如果读入的值为 0，则输出：

13. \* (方法覆盖)

```

import java.io.*;
class MySuper{
    public void m() throws IOException{}
}

```

```

class MySub extends MySuper{
    public void m() throws EOFException{}
}
class MySub2 extends MySub {
    public void m() throws FileNotFoundException{}
}

```

以上代码是否能编译通过？如果不能，应该如何修改？

14. \*（自定义异常）完成某个计费系统的用户登录和注册模块，要求如下：

1) 创建一个 `User` 类，包括：用户登录名（`username`）、密码（`password`）、用户真实姓名（`name`）、电子邮件地址（`email`）属性和相应的构造方法及 `set/get` 方法。

2) 创建两个自定义异常类，一个 `LoginException`，表示登录异常。一个 `RegisterException`，表示注册异常。自定义的两个异常，都要求有一个接受字符串类型参数的构造方法。

3) 创建一个 `UserBiz` 接口，该接口中定义两个方法：

```

void register(String username, String password, String password2,
String name, String email) throws RegisterException //用户注册

```

```

void login(String username, String password) throws LoginException //用户登录

```

其中 `register` 方法接受两个 `password` 参数，原因是：在用户注册时，需要输入两遍 `password`，只有两次输入的 `password` 一致，才允许注册。

4) 创建 `UserBiz` 接口的实现类。其中

为该实现类创建一个属性，该属性为一个 `Map`，用来保存已注册的用户信息。`Map` 的键为用户登录名，值为登录名对应的 `User` 对象。初始情况下，`Map` 中保存的对象为以下两个：

用户名	密码	真实姓名	电子邮件
admin	admin	Administrator	admin@123.com
Tom	cat	tomcat	tomcat@cat.com

`register` 方法在以下两种情况下抛出异常：

- 1) `username` 在 `Map` 中已存在
- 2) 两次输入的 `password` 不一致

`login` 方法在以下两种情况下抛出异常：

- 1) `username` 不存在
- 2) `username` 和 `password` 不匹配

5) 创建一个 `UserView` 接口，该接口中定义两个方法：

```

void login();

```

```

void register();

```

6) 创建 `UserView` 接口的实现类。

该实现类的 `login` 方法中，利用命令行，让用户输入用户名和密码，之后调用 `UserBiz` 中的 `login` 方法。部分代码如下：

```

void login(){
    System.out.println("请输入用户名: ");
    String username = /*读入用户名...*/;
}

```

```

System.out.println("请输入密码");
String password = /*读入密码*/;
//调用 UserBiz 中的 login 方法
}

```

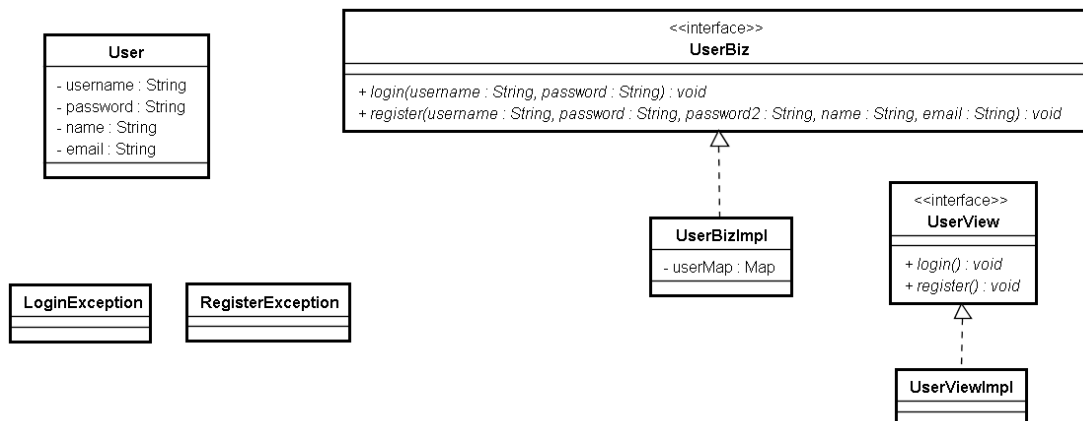
该类的 register 方法采用类似的方法，让用户输入注册时需要的信息，然后调用 UserBiz 中的 register 方法。

注意：

- 1、 密码应该让用户输入两遍。
- 2、 UserViewImpl 中应当有一个 UserBiz 类型的属性

7) 编写测试代码。

类图如下：



15. \*\*（异常的捕获和抛出）有以下代码：

```

import java.io.*;
import java.sql.*;
public class TestMyException{
    public static void main(String args[]){
        try{
            System.out.println("main1 ");
            ma();
            System.out.println("main2 ");
        }catch(Exception e){
            System.out.println("Catch Exception in main");
            System.out.println(e.getMessage());
        }
    }
    public static void ma() throws IOException{
        try{
            System.out.println("ma1 ");
            mb();
            System.out.println("ma2 ");
        }catch(SQLException e){

```



```

        System.out.println("Catch SQLException in ma");
        throw new IOException(e.getMessage());
    } catch (Exception e) {
        System.out.println("Catch Exception in ma");
        System.out.println(e.getMessage());
    }
}
}
public static void mb() throws SQLException {
    throw new SQLException("sql exception in mb");
}
}

```

问：该程序输出结果是什么？

16. \*\*（异常的捕获和抛出）有以下代码

```

public class TestException {
    public static void main(String args[]) {
        try {
            System.out.println("main1");
            ma();
            System.out.println("main2");
        } catch (Exception e) {
            System.out.println("In Catch");
        }
    }
    public static void ma() {
        System.out.println("ma1");
        throw new NullPointerException();
        System.out.println("ma2");
    }
}

```

选择正确答案：

- A. 编译出错
- B. 编译正常，输出 main1 ma1 In Catch
- C. 编译正常，运行时出错

17. \*\*（异常的捕获和抛出）有如下代码

```

import java.io.*;
import java.sql.*;
class TestException {
    public static void main(String args[]) {
        try {
            ma();
        }
        /*1*/
    }
}

```

```

        catch(Exception e){}
    }
    public static void ma() throws IOException{ }
}

```

下面哪些代码放在/\*1\*/处可以编译通过？

- A. catch(NullPointerException npe){}
- B. catch(IOException ioe){}
- C. catch(SQLException sqle){}

18. \*\*（finally）有如下代码

```

public class TestTryAndTry {
    public static void main(String args[]){
        System.out.println(ma());
    }
    public static int ma(){
        try{
            return 100;
        }finally{
            try{
                return 200;
            }finally{
                return 500;
            }
        }
        return 1000;
    }
}

```

选择正确答案：

- A. 编译错误
- B. 输出 100
- C. 输出 200
- D. 输出 500
- E. 输出 1000