Chp11 异常

参考答案

1. 参考答案

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

2. 参考答案

   异常类 java.rmi.AlreadyBoundException，从分类上说，该类属于已检查异常，从处理方式上说，对这种异常必须要处理；
   异常类 java.util.regex.PatternSyntaxException，从分类上说，该类属于未检查异常，从处理方式上说，对这种异常可处理可不处理。

3. 参考答案

```
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
            throw new NullPointerException();
        }else{
            //抛出一个 ClassCastException
            //并设定详细信息为“类型转换出错”
            throw new ClassCastException(“类型转换出错”);
        }
    }
}
```

4. 参考答案

   n = 1 时，输出

```
main1
ma1
mb1
Catch EOFException
In finally
```

```
    main2
n = 2 时，输出
    main1
    ma1
    mb1
    Catch IOException
    In finally
    main2
n = 3 时，输出
    main1
    ma1
    mb1
    Catch SQLException
    In finally
    main2
n = 4 时，输出
    main1
    ma1
    mb1
    Catch Exception
    In finally
    main2
n = 5 时，输出
    main1
    ma1
    mb1
    mb2
    ma2
    In finally
    main2
```

注意：不论是否出现异常，出现什么异常，In finally 语句都会被打印出来。

5. 参考第 6 题答案
6. 参考 TestMyException.java

7. 参考答案

```
//MyException 类必须要继承自 Throwable 的某一个子类
class MyException{}
class TestException{
    public static void main(String args[]){
        ma();
    }

    public static int ma(){
```

```
        try{
            m();
            return 100;
        }
        //捕获 Exception 的语句应当放在所有 catch 语句的最后。
        catch(Exception e){
            System.out.println("Exception");
        }
        catch(ArithmeticException e){
            System.out.println("ArithmeticException");
        }
    }
    //根据 MyException 是否继承自 RuntimeException
    //此处考虑是否声明抛出
    public static void m(){
        throw new MyException();
    }
}
```

8. 参考答案
   //1 处 A、B，//2 处 D。
   根据方法覆盖的要求，子类的覆盖方法不能比父类的被覆盖方法抛出更多的异常。ma 方法抛出 IOException，则子类可以抛出 IOException，也可以抛出 IOException 的子类；mb 方法没有抛出任何异常，则子类也不能抛出任何异常。

9. A
   由于 ma 方法声明抛出 Exception，因此在 main 方法中的 try 块必须有对 Exception 类型处理的子句。

10. A
    当程序异常时，返回值为-1，而当程序正常时，返回值为 n。问题在于，n 是一个局部变量，由于 try 块中的代码可能不会被执行，因此在 return 语句之前没有对 n 的赋值语句，这样就破坏了局部变量“先赋值，后使用”的要求，因此编译不通过。

11. 两次输出结果均为 100
    由于 finally 语句块中的代码一定要执行，因此，最后返回值一定是 finally 语句块中的返回值。

12. 当读入 10 时，输出为
    ma1
    ma2 1
    In Finally

    当读入 0 时，输出为

ma1
In Finally

13. 不能编译通过。由于 MySub2 继承自 MySub 类，因此不能抛出比 MySub 中的方法更多的异常。
由于 MySub 类中的 m 方法抛出 EOFException，而 MySub2 类的 m 方法抛出的 FileNotFoundException 不是 EOFException 的子类，因此这个方法抛出了比 MySub 类中的 m 方法更多的异常，因此编译不通过。

14. 参考 TestException.java 文件

15. 程序输出的结果如下：
```
main1
ma1
Catch SQLException in ma
Catch Exception in main
sql exception in mb
```
描述如下：
首先 `main1` 和 `ma1` 是正常执行流程输出的结果；
之后，`ma` 方法中调用 `mb` 方法，`mb` 方法抛出一个 `SQLException`，该异常被 `ma` 方法的 `catch(SQLException e)` 捕获；
在 `ma` 的 `catch` 语句中，输出 `Catch SQLException in ma`，然后重新抛出一个 `IOException`，注意，此时该异常向 `ma` 方法的调用者抛出；
在 `main` 方法的 `catch` 语句中捕获到 `ma` 方法抛出的异常，因此输出 `Catch Exception in main`，之后输出异常的详细信息：`sql exception in mb`

16. A
ma 方法中第二个输出语句，由于上一个语句是 throw 语句，因此第二个输出语句永远都执行不到，因此编译出错。

17. AB
在/*1*/处，由于 ma 方法声明有可能抛出 IOException 异常，因此 B 编译通过；
由于 NullPointerException 异常是未检查异常，即使 ma 方法的 throws 语句中没有声明抛出，调用时也有可能产生该异常，因此 A 编译通过；
由于 SQLException 没有声明抛出，并且是已检查异常，因此在/*1*/处不可能捕获到该类异常，因此 C 编译不通过

18. A
这个题目的难点在于 finally 语句中还套着一个 try-finally 结构。
对于外层的 try-finally 结构而言，finally 语句一定要执行，因此返回值一定是 finally 语句块中的某个值；
对于内层的 try-finally 结构而言，finally 语句一定要执行，因此返回值一定是 500；
而在 return 500 后面的语句，一定不会被执行，因此编译不通过。