1.体系结构

```
java.lang.Object
    |----java.lang.Throwable
            |-------java.lang.Error：错误，java程序对此无能为力，不显式的处理
            |-------java.lang.Exception:异常。需要进行处理
                    |------RuntimeException:运行时异常
                            |-----ArrayIndexOutOfBoundsException/NullPointerException/ArithmeticException/ClassCastException
                    |------非RuntimeException:编译时异常
```

2.因为java程序分为javac.exe和java.exe两个过程，在每个过程中，都有可能出现异常。故分为编译时异常、运行时异常

2.1 对于运行时异常比较常见，可以不显式的来处理。

2.2 对于编译时异常，必须要显式的处理

编译时异常，不是说有异常才处理，而是存在异常的隐患，必须在编译前，提示程序，万一出现异常，如何处理！

处理的方式一：

```
try{
  //可能出现异常的代码
}catch(Exception1 e1){
  //处理的方式1
}catch(Exception2 e2){
  //处理的方式2
}finally{
  //一定要执行的代码
}
```

注：1.try内声明的变量，类似于局部变量，出了try{}语句，就不能被调用

2.finally是可选的。

3.catch语句内部是对异常对象的处理：

>getMessage(); printStackTrace();

4.可以有多个catch语句，try中抛出的异常类对象从上往下去匹配catch中的异常类的类型，一旦满足就执行catch中的代码。执行完，就跳出其后的多条catch语句

5.如果异常处理了，那么其后的代码继续执行。

6.若catch中多个异常类型是"并列"关系，孰上孰下都可以。

若catch中多个异常类型是"包含"关系，须将子类放在父类的上面，进行处理。否则报错！

7.finally中存放的是一定会被执行的代码，不管try中、catch中是否仍有异常未被处理，以及是否有return语句。

8.try-catch是可以嵌套的。

处理方式二：

在方法的声明处，显式的使用throws + 异常类型

编译异常一定要进行处理

1，手动的抛出一个异常，可以使用throw+异常类对象，来手动抛出一个异常

2，如何自定义一个异常类：

手动的抛出一个异常，除了抛出的是现成的异常类对象之外，还可以抛出一个自定义的异常对象

```
>如何自定义一个异常类呢？
//1.自定义的异常类继承现有的异常类
//2.提供一个序列号，提供几个重载的构造器
public class MyException extends Exception{

    static final long serialVersionUID = -70348975766939L;

    public MyException(){

    }
    public MyException(String msg){
        super(msg);
    }
}
```

3，五个关键字搞定异常：“”其中要区分throw与throws的区别

集合：①数组：存储基本数据类型和引用数据类型②集合只能存储引用数据类型

数组存储数据的弊端：声明数组的时候就需要知道数组的大小，数组中真正存储了多少个元素

不得而知

集合框架：

vector是线程安全的