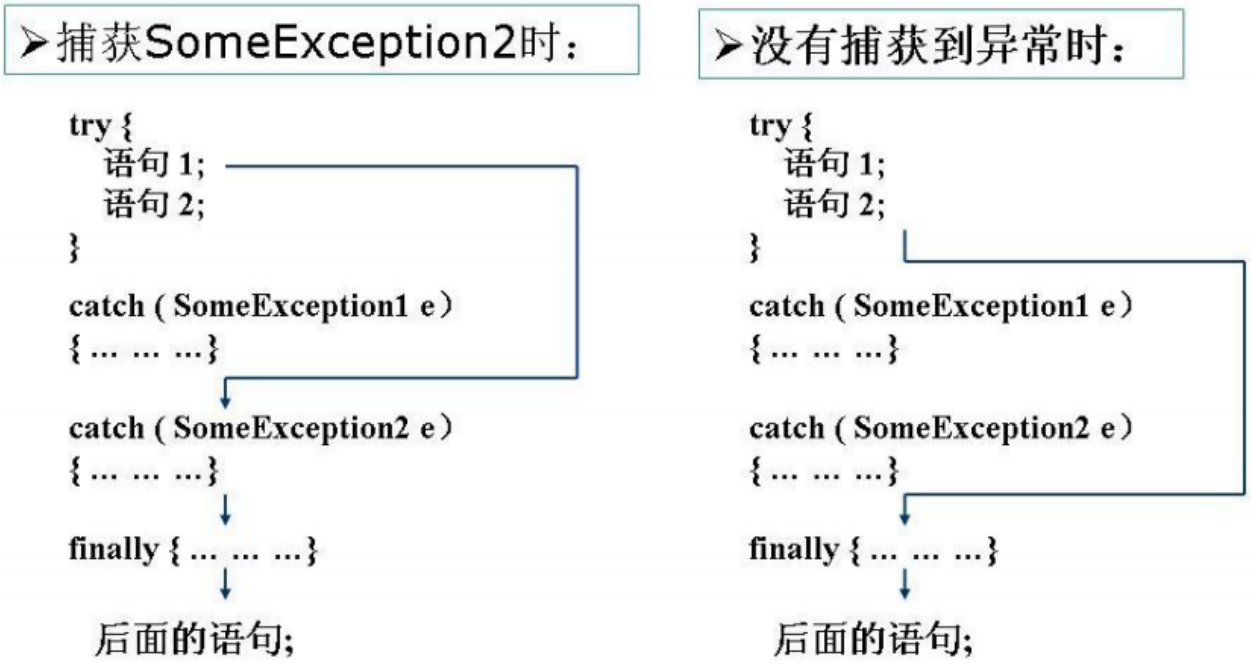
异常处理

异常的概念

- java异常是java提供的用于处理程序中错误的一种机制
- 所谓错误是指程序运行过程中发生的一些异常事件（除0、数组下标越界、读取的文件不存在）
- java程序运行过程中如出现异常事件，可以生产一个异常类对象，该异常对象封装了异常事件的信息并提交给java运行系统，这个过程称为抛出异常
- 当java运行时，系统接收到异常对象时，会寻找能处理异常的代码，并把当前异常对象交给其处理，这一过程称为捕获异常

异常的分类

- ERROR：称为错误，有java虚拟机生成并抛出，包括动态链失败，虚拟机错误等，程序不对其处理
- Exception：所以异常的父类，其子类对应了各种各样可能出现的异常事件，一般需要用户显示声明或捕获
    - RuntimeException：一类特殊的异常，通常包括
        - ArithmeticException、ArrayStoreException、IndexOutOfBoundsException、NullPointerException
    - IOException、classNotFoundException、NoSuchMethodException



方法的异常自己处理

```
package 异常;

import org.junit.Test;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
```

```
public class test02 {

    public void function1(){
        FileInputStream in = null;
        try{
            in = new FileInputStream("myprofile.txt");

        }catch (FileNotFoundException e){
            System.out.println(e.getMessage());
            e.printStackTrace();
        }catch (IOException e){
            e.printStackTrace();
        }finally {
            try {
                in.close();
            }catch (IOException e){
                e.printStackTrace();
            }
        }
    }

    @Test
    public void function2(){
        function1();
    }
}
```

方法的异常抛出，交给调用该方法的函数处理

```
package 异常;

import org.junit.Test;

import java.io.IOException;

public class test01 {
    public static void main(String[] args) {

    }

    public void fun(int i) throws IOException,ArithmeticException{
        if (i == 0)
            throw new ArithmeticException("被除数为0");
        if (i == 1)
            throw new IOException("输入输出流异常");
    }

    @Test
    public void test(){
```

```
        try {
            fun(1);
        } catch (IOException e) {
            System.out.println( e.getMessage());
            e.printStackTrace();
        }catch (ArithmeticException e){
            System.out.println(e.getMessage());
        }finally {
            System.out.println("执行finally");
        }
    }
}
```

自定义异常类

- 通过继承java.lang.Exception类声明该类为异常类

```
package 异常;

import org.junit.Test;

class MyException extends Exception{
    private int id;
    MyException(String message,int id){
        super(message);
        this.id = id;
    }
    public int getErrorId(){
        return id;
    }
}
public class test03 {

    public void regist1(int num){
        try {
            if (num < 0){
                throw new MyException("注册人数不能为负数",400);
            }
            else
                System.out.println("注册成功");
        }catch (MyException e){
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }

    @Test
    public void test(){
        regist1(200);
        regist1(-1);
```

```
    }
}
```

声明并抛出异常

- 如果不是RuntimeException，重写的方法需要与原方法抛出的异常类型一致或者不抛出异常。
- 子类抛出RuntimeException异常不受父类限制，因为RuntimeException可以抛也可以不抛

```
package 异常;

import java.io.IOException;

class A{
    public void fun() throws IOException{


    }
}
class B extends A{
    public void fun(){

    }
}
class C extends A{
    public void fun() throws IOException{

    }
}
class D extends A {
    public void fun() throws NullPointerException{

    }
}
public class test04 {

}
```

继承类之间的异常嵌套

```
package 异常;

import org.junit.Test;

class FatherException extends Exception{
    public void print(){
        System.out.println("FarherException");
    }
```

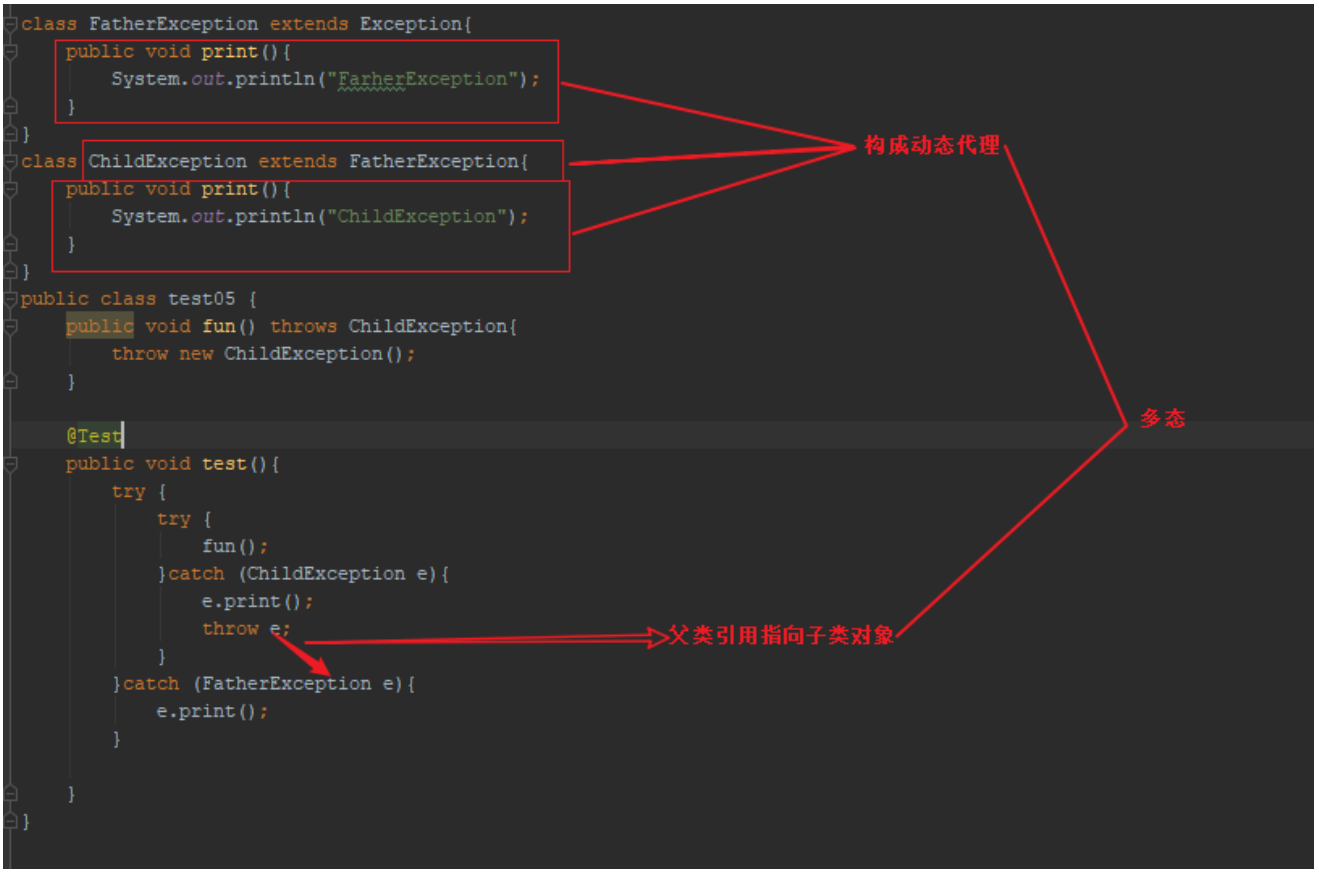
```
}
class ChildException extends FatherException{
    public void print(){
        System.out.println("ChildException");
    }
}
public class test05 {
    public void fun() throws ChildException{
        throw new ChildException();
    }

    @Test
    public void test(){
        try {
            try {
                fun();
            }catch (ChildException e){
                e.print();
                throw e;
            }
        }catch (FatherException e){
            e.print();
        }

    }
}
```

结果：

```
    ChildException
    ChildException
```



总结：

- 一个异常图
- 五个关键字
  - try
  - catch
  - throw
  - finally
  - throws

- 先逮小的，在逮大的
- 异常和重写的关系