# 实验六 异常

【实验目的】

1. 掌握异常的概念和处理。
2. 掌握自定义异常创建与应用。

【实验要求】

1. 利用异常处理机制处理程序中的异常。

2. 自定义异常类并应用其处理程序中的异常。

【实验内容】

1. 自定义异常类MyException，该类继承自Exception类，类中只有含一个字符串参数msg的构造方法，构造方法中只有一条语句super(msg)——调用父类的构造方法。另外，编写自定义类person，类中只有两个私有的变量，一个是字符串类型的姓名，另一个是整型变量age；有两个公有方法getAge()和void setAge(int age)，其中setAge(int age)的功能是把参数age的值加到类中的变量age中(但要求age>0，否则抛出自定义异常MyException类的对象)，getAge()方法返回age的值。编写应用程序。

\_public class Person {  
 String name;  
 Integer age;  
  
 public Integer getAge() {  
 return age;  
 }  
  
 public void setAge(Integer age) throws MyException {  
 if (age <= 0) throw new MyException("Age is invalided.");  
 this.age = age;  
 }  
  
}

public class MyException extends Exception {  
 String msg;  
  
 public MyException(String msg) {  
 super(msg);  
 }  
}

2.练习教材上异常相关的程序

public interface BaseEnum {  
 Integer getCode();  
 String getValue();  
}

public enum ExpectionEnum implements BaseEnum {  
 *NAME\_SIZE\_SIZE\_ERROR*(1001, "Name is too long."),  
 *NAME\_SIZE\_NULL\_ERROR*(1002, "Name is empty."),  
 *AGE\_RANGE\_ERROR*(1003, "The range of age is invalid.")  
 ;  
  
 private Integer code;  
 private String value;  
  
 ExpectionEnum(Integer code, String value) {  
 this.code = code;  
 this.value = value;  
 }  
  
 @Override  
 public Integer getCode() {  
 return this.code;  
 }  
  
 @Override  
 public String getValue() {  
 return this.value;  
 }  
}

public class StuExpection extends Exception {  
 private Integer code;  
 private String msg;  
  
 public StuExpection(ExpectionEnum e) {  
 super(e.getValue());  
 this.code = e.getCode();  
 this.msg = e.getValue();  
 }  
}

public class Student {  
 private String name;  
 private Integer age;  
  
 public Student(String name, Integer age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 @Override  
 public String toString() {  
 return "name: " + name + " age: " + age;  
 }  
}

public class StudentTest {  
 public static void main(String[] args) throws StuExpection {  
 Scanner scanner = new Scanner(System.*in*);  
 String name = scanner.nextLine();  
 int age = scanner.nextInt();  
 if (name.isEmpty()) throw new StuExpection(ExpectionEnum.*NAME\_SIZE\_NULL\_ERROR*);  
 if (name.length() > 5) throw new StuExpection(ExpectionEnum.*NAME\_SIZE\_SIZE\_ERROR*);  
 if (age <= 0 || age > 110) throw new StuExpection(ExpectionEnum.*AGE\_RANGE\_ERROR*);  
 Student student = new Student(name, age);  
 System.*out*.println(student);  
 }  
}

【实验成绩】