## 面向接口编程的实际应用

面向接口编程是一种常用的编程方式，可以有效地提高代码的复用性，增强程序的扩展性和维护性。

- 场景

某工厂生产产品 A，产品 A 主要是由设备 A 来完成生产，用程序模拟生产过程。

产品 B 是由设备 B 来生产的。

当需求发生改变时，就要频繁修改类的内部结构的方式是需要避免的，因为这种结构的程序扩展性很差，如何改进？使用面向接口编程即可。

1、创建接口 Equiment。

```
package com.southwind.test;

public interface Equiment {
  public void work();
}
```

2、创建 Equiment 的实现类。

```
package com.southwind.test;

public class EquipmentA implements Equiment {
  public void work() {
    System.out.println("设备A运行，生产产品A");
  }
}

package com.southwind.test;

public class EquipmentB implements Equiment {
  public void work() {
    System.out.println("设备B运行，生产产品B");
  }
}

package com.southwind.test;

public class EquipmentC implements Equiment {
  public void work() {
    // TODO Auto-generated method stub
    System.out.println("设备C运行，生产产品C");
  }

}
```

3、创建 Factory 类。

```
package com.southwind.test;

public class Factory {
  private Equiment equipment;

  public Equiment getEquipment() {
    return equipment;
  }

  public void setEquipment(Equiment equipment) {
    this.equipment = equipment;
  }

  public void work() {
    System.out.println("开始生产...");
    this.equipment.work();
  }
}
```

4、Test

```
package com.southwind.test;

public class Test {
  public static void main(String[] args) {
    //初始化工厂
    Factory factory = new Factory();
//    EquipmentB equipmentB = new EquipmentB();
//    factory.setEquipment(equipmentB);
//    EquipmentA equipmentA = new EquipmentA();
//    factory.setEquipment(equipmentA);
    EquipmentC equipmentC = new EquipmentC();
    factory.setEquipment(equipmentC);
    //开始工作
    factory.work();
  }
}
```

## 异常

- 什么是异常?

Java 中的错误大致可以分为两类：

一类是编译时错误，一般是指语法错误。

另一类是运行时错误。

Java 中有一组专门用来描述各种不同的运行时异常，叫做异常类，Java 结合异常类提供了处理错误的机制。

具体步骤是当程序出现错误时，会创建一个包含错误信息的异常类的实例化对象，并自动将该对象提交给系统，由系统转交给能够处理异常的代码进行处理。

异常可以分为两类：Error 和 Exception。

Error 是指系统错误，JVM 生成，我们编写的程序无法处理。

Exception 指程序运行期间出现的错误，我们编写的程序可以对其进行处理。

- 异常的使用

异常的使用需要用到两个关键字 try 和 catch，并且这两个关键字需要结合起来使用，用 try 来监听可能会抛出异常的代码，一旦捕获到异常，生成异常对象并交给 catch 来处理，基本语法如下所示。

```
try{
  //可能抛出异常的代码
}catch(Exception e){
  //处理异常
}
```

```
package com.southwind.exception;

public class Test {
  public static void main(String[] args) {
    try {
      int num = 10/10;
    }catch (Exception e) {
      // TODO: handle exception
      if(e.getMessage().equals("/ by zero")) {
        System.err.println("分母不能为0");
      }
    }
  }
}
```

除了 try 和 catch，还可以使用 finally 关键字来处理异常，finally 的作用?

无论程序是否抛出异常，finally 代码块中的代码一定都会执行，finally 一般跟在 catch 代码块的后面，基本语法如下所示。

```
try{
  //可能抛出异常的代码
}catch(Exception e){
  //处理异常
}finally{
  //必须执行的代码
}
```

- 异常类

Java 将运行时出现的错误全部封装成类，并且不是一个类，而是一组类。同时这些类之间是有层级关系的，由树状结构一层层向下分级，处在最顶端的类是 Throwable，是所有异常类的根结点。

Throwable 有两个直接子类：Error 和 Exception。

Error 的子类包括 VirtualMachineError、AWTError、IOError。

VirtualMachineError 的子类有 StackOverflowError、OutOfMemoryError。

Exception 的子类包括 IOException 和 RuntimeException。

IOException 的子类包括 FileLockInterruptionException、FileNotFoundException、FilerException。

RuntimeException 的子类包括 ArithmeticException、ClassNotFoundException、IllegalArggumentException、ArrayIndexOutOfBoundsException、NullPointerException、NoSuchMethodException、NumberFormatException。