链滴

一个实例理解 Java 的接口（interface）用处与用法

作者：adlered

原文链接：https://ld246.com/article/1552317815323

来源网站：链滴

最近突然想到一个问题：Runnable是如何允许我自定义执行内容而进行多线程处理的？
首先看使用Runnable创建多线程的方法：

```
Runnable runnable = new Runnable() {
    @Override
    public void run() {
        System.out.println("hello~");
    }
};
Thread thread = new Thread(runnable);
thread.run();
```

我们可以看到首先创建了一个Runnable接口实例，Runnable接口源代码如下：

```
package java.lang;
/*
 * @author  Arthur van Hoff
 * @see     java.lang.Thread
 * @see     java.util.concurrent.Callable
 * @since   JDK1.0
*/
public interface Runnable {
  public abstract void run();
}
```

原文链接：一个实例理解 Java 的接口（interface）用处与用法

可以看到的是在Runnable接口中只有一个run方法待用户定义。

当我们重写Runnable的run方法后，我们就有了一个完整的方法，然后通过Thread传入接口并使用ru();方法运行。

所以**Thread**包括了多线程执行的所有方法，而**Runnable**仅提供了一个供用户重写的接口。

回过头来，我们创建一个实例：

- Main.java
- TestIF.java

TestIF.java

该接口仅负责存储一个String字符串：

```
public interface TestIF {
    String str();
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        //创建一个TestIF接口实例并返回
        TestIF testIF = IF();
        //传递给另一个方法并输出
        PrintIF(testIF);
    }

    public static TestIF IF() {
        TestIF testIF = new TestIF() {
            @Override
            public String str() {
                return "SUCCESSFUL!";
            }
        };
        return testIF;
    }

    public static void PrintIF(TestIF testIF) {
        System.out.println("Has received: " + testIF.str());
    }
}
```

输出结果

```
Has received: SUCCESSFUL!
```

总结

来自jiany的订正：

抽象是模板，接口是契约。接口规定了这个类可以实现的功能。

原文链接：一个实例理解 Java 的接口（interface）用处与用法