# Java 9 私有接口方法

在 Java 8之前，接口可以有常量变量和抽象方法。

我们不能在接口中提供方法实现。如果我们要提供抽象方法和非抽象方法（方法与实现）的组合，那么我们就得使用抽象类。

## 实例

public class Tester { public static void main(String []args) { LogOracle log = new LogOracle(); log.logInfo(""); log.logWarn(""); log.logError(""); log.logFatal(""); LogMySql log1 = new LogMySql(); log1.logInfo(""); log1.logWarn(""); log1.logError(""); log1.logFatal(""); } } final class LogOracle implements Logging { @Override public void logInfo(String message) { getConnection(); System.out.println("Log Message : " + "INFO"); closeConnection(); } @Override public void logWarn(String message) { getConnection(); System.out.println("Log Message : " + "WARN"); closeConnection(); } @Override public void logError(String message) { getConnection(); System.out.println("Log Message : " + "ERROR"); closeConnection(); } @Override public void logFatal(String message) { getConnection(); System.out.println("Log Message : " + "FATAL"); closeConnection(); } @Override public void getConnection() { System.out.println("Open Database connection"); } @Override public void closeConnection() { System.out.println("Close Database connection"); } } final class LogMySql implements Logging { @Override public void logInfo(String message) { getConnection(); System.out.println("Log Message : " + "INFO"); closeConnection(); } @Override public void logWarn(String message) { getConnection(); System.out.println("Log Message : " + "WARN"); closeConnection(); } @Override public void logError(String message) { getConnection(); System.out.println("Log Message : " + "ERROR"); closeConnection(); } @Override public void logFatal(String message) { getConnection(); System.out.println("Log Message : " + "FATAL"); closeConnection(); } @Override public void getConnection() { System.out.println("Open Database connection"); } @Override public void closeConnection() { System.out.println("Close Database connection"); } } interface Logging { String ORACLE = "Oracle\_Database"; String MYSQL = "MySql\_Database"; void logInfo(String message); void logWarn(String message); void logError(String message); void logFatal(String message); void getConnection(); void closeConnection(); }

以上实例执行输出结果为：

Open Database connectionLog Message : INFOClose Database connectionOpen Database connectionLog Message : WARNClose Database connectionOpen Database connectionLog Message : ERRORClose Database connectionOpen Database connectionLog Message : FATALClose Database connection

在上面的例子中，每个日志方法都有自己的实现。

在 Java 8 接口引入了一些新功能——默认方法和静态方法。我们可以在Java SE 8的接口中编写方法实现，仅仅需要使用 **default** 关键字来定义它们。

在 Java 8 中，一个接口中能定义如下几种变量/方法：

* **常量**
* **抽象方法**
* **默认方法**
* **静态方法**

## 实例

public class Tester { public static void main(String []args) { LogOracle log = new LogOracle(); log.logInfo(""); log.logWarn(""); log.logError(""); log.logFatal(""); LogMySql log1 = new LogMySql(); log1.logInfo(""); log1.logWarn(""); log1.logError(""); log1.logFatal(""); } } final class LogOracle implements Logging { } final class LogMySql implements Logging { } interface Logging { String ORACLE = "Oracle\_Database"; String MYSQL = "MySql\_Database"; default void logInfo(String message) { getConnection(); System.out.println("Log Message : " + "INFO"); closeConnection(); } default void logWarn(String message) { getConnection(); System.out.println("Log Message : " + "WARN"); closeConnection(); } default void logError(String message) { getConnection(); System.out.println("Log Message : " + "ERROR"); closeConnection(); } default void logFatal(String message) { getConnection(); System.out.println("Log Message : " + "FATAL"); closeConnection(); } static void getConnection() { System.out.println("Open Database connection"); } static void closeConnection() { System.out.println("Close Database connection"); } }

以上实例执行输出结果为：

Open Database connectionLog Message : INFOClose Database connectionOpen Database connectionLog Message : WARNClose Database connectionOpen Database connectionLog Message : ERRORClose Database connectionOpen Database connectionLog Message : FATALClose Database connection

Java 9 不仅像 Java 8 一样支持接口默认方法，同时还支持私有方法。

在 Java 9 中，一个接口中能定义如下几种变量/方法：

* **常量**
* **抽象方法**
* **默认方法**
* **静态方法**
* **私有方法**
* **私有静态方法**

以下实例提取了冗余到通用方法，看起来明显更简洁：

## 实例

public class Tester { public static void main(String []args) { LogOracle log = new LogOracle(); log.logInfo(""); log.logWarn(""); log.logError(""); log.logFatal(""); LogMySql log1 = new LogMySql(); log1.logInfo(""); log1.logWarn(""); log1.logError(""); log1.logFatal(""); } } final class LogOracle implements Logging { } final class LogMySql implements Logging { } interface Logging { String ORACLE = "Oracle\_Database"; String MYSQL = "MySql\_Database"; private void log(String message, String prefix) { getConnection(); System.out.println("Log Message : " + prefix); closeConnection(); } default void logInfo(String message) { log(message, "INFO"); } default void logWarn(String message) { log(message, "WARN"); } default void logError(String message) { log(message, "ERROR"); } default void logFatal(String message) { log(message, "FATAL"); } private static void getConnection() { System.out.println("Open Database connection"); } private static void closeConnection() { System.out.println("Close Database connection"); } }

以上实例执行输出结果为：

Open Database connectionLog Message : INFOClose Database connectionOpen Database connectionLog Message : WARNClose Database connectionOpen Database connectionLog Message : ERRORClose Database connectionOpen Database connectionLog Message : FATALClose Database connection