

**编制：竹子**

Java Design pattern

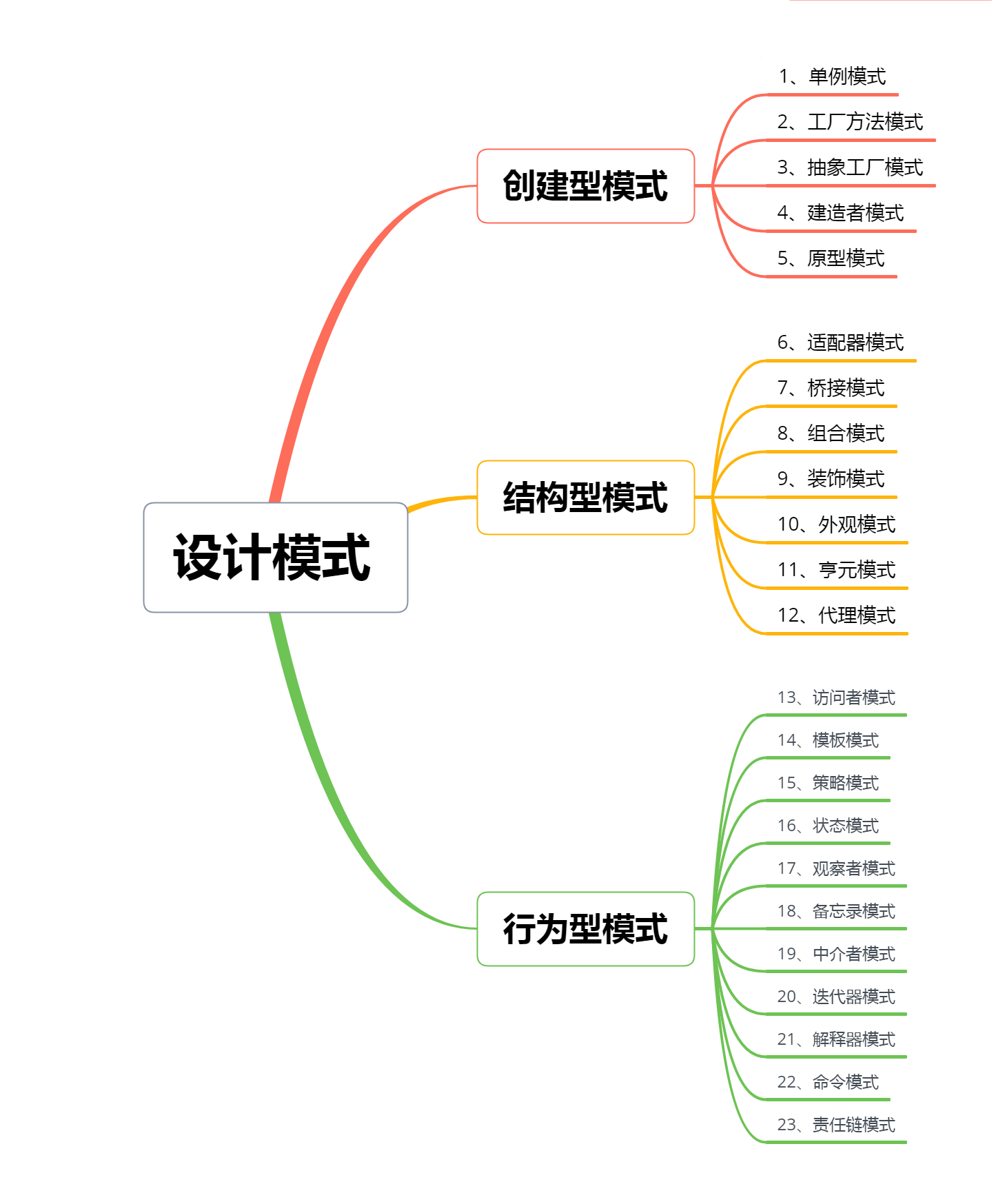
**设计模式**

# **一、设计模式分类**

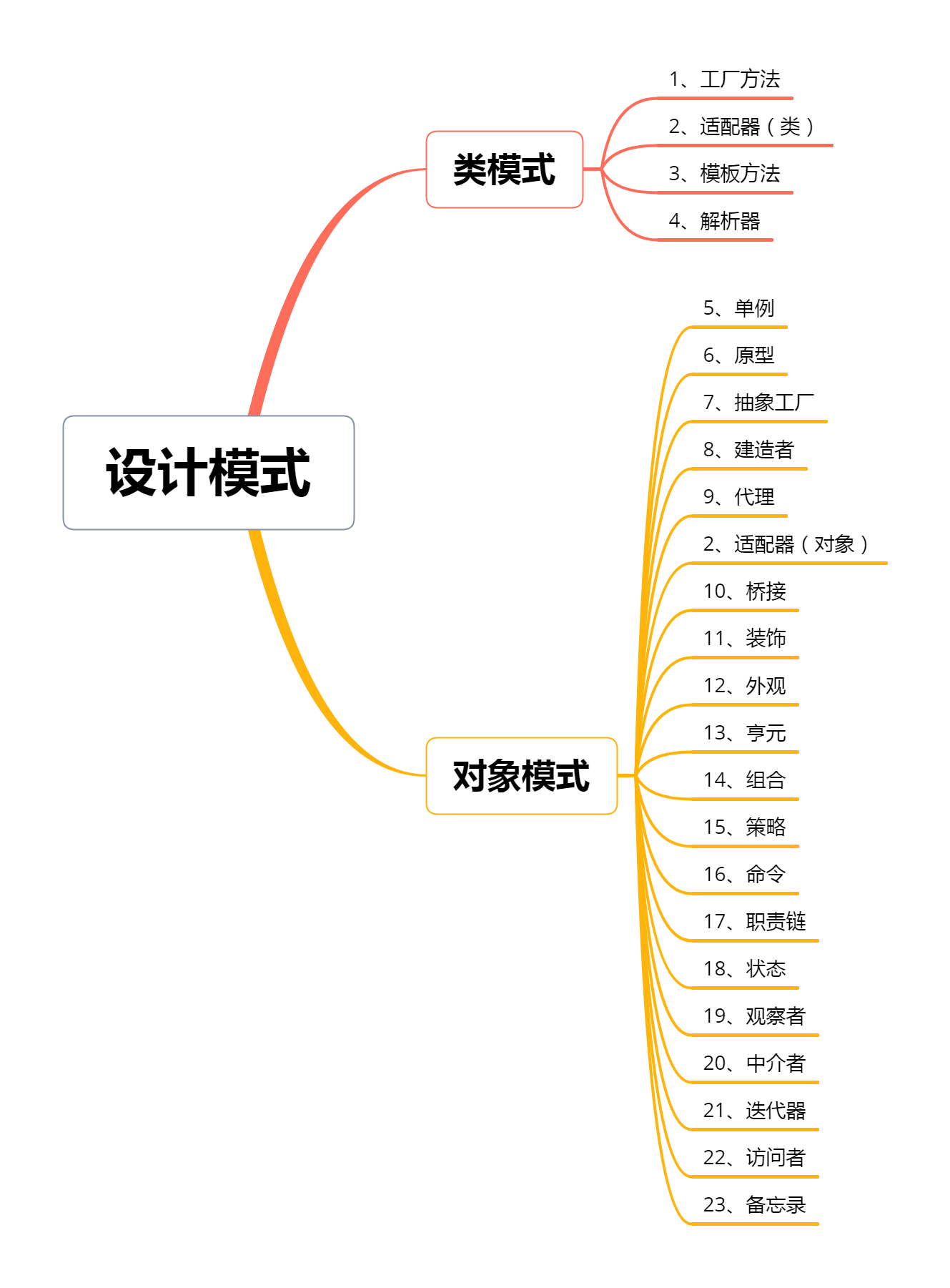
## **1.1.GOF分类表格**

|  |  |  |  |
| --- | --- | --- | --- |
| 范围\目的 | 创建型模式 | 构造型模式 | 行为型模式 |
| 类模式 | 工厂方法 | 适配器（类） | 模板方法  解析器 |
| 对象模式 | 单例  原型  抽象工厂  建造者 | 代理  适配器（对象）  桥接  装饰  外观  亨元  组合 | 策略  命令  职责链  状态  观察者  中介者  迭代器  访问者  备忘录 |

## **1.2.根据目的分类**



## **1.3.根据范围分类**



# **二、定义和应用**

## **2.1.单例模式**

**定义**：某个类只能有一个实例，并提供一个全局的访问点获取该实例。

**特点：**1、只有一个实例对象。 2、自我实例化。 3、提供全局访问点。

**举例：**Windows 中只能打开一个任务管理器

### 2.1.1.第一种写法：懒汉式线程不安全单例模式

#### 2.1.1.1.单例类代码

|  |
| --- |
| 单例类 |
| /\*\*  \* 懒汉式线程不安全单例模式  \* **@author** 竹子  \* **@version** 1.0  \*/ **public class LazySingleTon** {   /\*\* 1.只有一个实例对象 \*/  **private static LazySingleTon instance**;   /\*\* 2.自我实例化 \*/  **private LazySingleTon**() {  **if** (**instance** != **null**) {  **throw new** IllegalStateException(**"实例已经初始化！"**);  }  }  /\*\* 3.提供全局访问点 \*/  **public static LazySingleTon** getInstance() {  **if** (**instance** == **null**) {  **instance** = **new** LazySingleTon();  }  **return instance**;  } } |

#### 2.1.1.2.测试类代码

|  |
| --- |
| 测试类 |
| /\*\*  \* 测试类  \* **@author** 竹子  \* **@version** 1.0  \*/ **public class Main** {   **public static void** main(**String**[] *args*) {  **LazySingleTon** ton1 = **LazySingleTon**.**getInstance**();  **System**.***out***.println(ton1);  **LazySingleTon** ton2 = **LazySingleTon**.**getInstance**();  **System**.***out***.println(ton2);  **System**.***out***.println(ton1 == ton2);  **System**.***out***.println(ton2.equals(ton1));  } } |

#### 2.1.1.3.说明

为什么说这是懒汉式呢？

因为实例对象只有在获取的时候才确定要不要实例化。

为什么说这是线程不安全呢？

因为提供实例获取的方法中，只是用了if (instance == null) 来确定要不要实例化，假设有两个线程去访问该方法，有可能会存在都执行instance = new LazySingleTon(); 这一句，所有就不能保证“只有一个实例对象”的原则了。所以说线程不安全。生产中建议不要使用“懒汉式线程不安全单例模式”。

### 2.1.2.第一种写法：懒汉式线程安全单例模式

#### 2.1.2.1.单例类代码

|  |
| --- |
| 单例类 |
| /\*\*  \* 懒汉式线程安全单例模式  \* **@author** 竹子  \* **@version** 1.0  \*/ **public class LazySingleTonSafe** {  /\*\* 1.只有一个实例对象 \*/  **private static LazySingleTonSafe instance**;  /\*\* 2.自我实例化 \*/  **private LazySingleTonSafe**() {  **if** (**instance** != **null**) {  **throw new** IllegalStateException(**"实例已经初始化！"**);  }  }  /\*\* 3.提供全局访问点 \*/  **public static synchronized LazySingleTonSafe** getInstance() {  **if** (**instance** == **null**) {  **instance** = **new** LazySingleTonSafe();  }  **return instance**;  } } |

#### 2.1.2.2.测试类代码

|  |
| --- |
| 测试类 |
| /\*\*  \* 测试类  \* **@author** 竹子  \* **@version** 1.0  \*/ **public class Main** {   **public static void** main(**String**[] *args*) {  **LazySingleTonSafe** ton1 = **LazySingleTonSafe**.**getInstance**();  **System**.***out***.println(ton1);  **LazySingleTonSafe** ton2 = **LazySingleTonSafe**.**getInstance**();  **System**.***out***.println(ton2);  **System**.***out***.println(ton1 == ton2);  **System**.***out***.println(ton2.equals(ton1));  } } |

#### 2.1.2.3.说明

在“懒汉式线程不安全单例模式”基础上，在获取实例对象的方法上加了synchronized。

避免了线程冲突，保持单个对象。