设计模式之创建模式 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/167/2021_2022__E8_AE_BE_E8_AE_A1_E6_A8_A1_E5_c104_167293.htm 1 Creational Patterns 将对象的使用与对象的创建分开。对象的使用者不负责创建对象，只需知道如何获取需 要的对象。这样，当被使用对象的创建过程需要变更或扩展时，对象的使用者无须变动，只要对专门负责创建该对象的对象工厂做必要的变更或扩展即可。 1.1 The Factory Pattern 1.1.1 基本类图 1.1.2 分析 当 BasicProduct 有新的子产品 ExProduct3 要应用到系统时，Factory 的 createProduct() 需要简单修改，增加一个“ else if ”分支，没有完全实现“ 对扩展开放，对修改封闭 ”。但当明确产品就是有限的几种时，仍是有效且常用的。 1.2 The Abstract Factory Pattern 1.2.1 基本类图 1.2.2 分析 当 BasicProduct 有新的子类 ExProduct3 要应用到系统时，新建一个对应的扩展自 BasicFactory 的 ExFactory3 即可，现有类无需修改，实现了“ 对扩展开放，对修改封闭 ”。 1.3 The Singleton Pattern 控制系统最多存在某个类的一个实例。很容易将 Singleton Pattern 扩展为控制系统只能 创建某个类的有限个实例的情形。系统中大部分管理服务对象都是单实例的。例如，系统存在多个用户实例，但只存在一个用户管理者对象。 1.3.1 常用实现方式 public class Singleton { private static Singleton instance = null . private Singleton() {}. // 控制外部不能自行创建 Sigleton 实例 public static Singleton getInstance() { if (null == instance) { instance = new Singleton(). } return instance . } } 1.4 The Builder Pattern 1.4.1 基本类图 1.4.2 分析 当一个产品（ Product ）由多

个部分（Part1， Part2，…， Partn）组成，而各个部分又有 不同的构造方式时。将产品各部分的构造和产品的组装分离，就是 Builder Pattern 。 Builder 是对产品各部分构造的抽象，而 Director 负责使用指定的 Builder 组装产品。这样，当系统有新的产品各部分构造方式需要加入时，只需实现继承自 Builder 的 ConcreteBuilder3 即可，系统其他类无需修改，实现了“ 对扩展开放，对修改关闭 ”。 1.5 The Prototype Pattern 给定一个对象实例 A，得到一个另一个对象实例 B ： B 的类型与 A 同， B 的内容与 A 相同。常称 B 为 A 的克隆或副本。 Java 对象模型直接支持 Prototype Pattern ，因为 Java 自带克隆机制。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com