Prototype Pattern
原型模式

Tom

原型模式的定义

原型模式（Prototype Pattern）是指原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

调用者不需要知道任何创建细节，不调用构造函数。

属于创建型模式

原型模式的适用场景

1、类初始化消耗资源较多。

2、new产生的一个对象需要非常繁琐的过程（数据准备、访问权限等）

3、构造函数比较复杂。

4、循环体中生产大量对象时。

浅克隆

深克隆

原型模式的优点

性能优良，Java自带的 原型模式 是基于内存二进制流的拷贝，比直接new一个对象性能上提升了许多。

可以使用深克隆方式保存对象的状态，使用原型模式将对象复制一份并将其状态保存起来，简化了创建过程

原型模式的缺点

必须配备克隆（或者可拷贝）方法

当对已有类进行改造的时候，需要修改代码，违反了开闭原则。

深拷贝、浅拷贝需要运用得当

Builder Pattern
建造者模式

Tom

建造者模式的定义

建造者模式（Builder Pattern）是将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

特征：用户只需指定需要建造的类型就可以获得对象，建造过程及细节不需要了解。

属于创建型模式。

建造者模式的适用场景

适用于创建对象需要很多步骤，但是步骤的顺序不一定固定。

如果一个对象有非常复杂的内部结构（很多属性）

把复杂对象的创建和使用分离

建造者模式的优点

封装性好，创建和使用分离

扩展性好，建造类之间独立、一定程度上解耦

建造者模式的缺点

产生多余的Builder对象

产品内部发生变化，建造者都要修改，成本较大

建造者模式和工厂模式的区别

1、建造者模式更加注重方法的调用顺序，工厂模式注重于创建对象。

2、创建对象的力度不同，建造者模式创建复杂的对象，由各种复杂的部件组成，工厂模式创建出来的都一样。

3、关注点：工厂模式模式只需要把对象创建出来就可以了，而建造者模式中不仅要创建出这个对象，还要知道这个对象由哪些部件组成。

4、建造者模式根据建造过程中的顺序不一样，最终的对象部件组成也不一样。