|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S | O | L | I | D |
| Single Responsibility Principle | **Open/Closed Principle** | **Liskov Substitution Principle** | **Interface Segregation Principle** | **Dependency Inversion Principle** |
| Princípio Responsabilidade Única | **Princípio Aberto/Fechado** | **Princípio de Substituição de Liskov** | **Princípio da Segregação de Interfaces** | **Princípio da Invasão de Dependência** |
| Cada classe deve ter uma única responsabilidade. | **As classes devem ser abertas para extensão, mas fechadas para modificação.** | **Objetos de uma classe base devem ser substituíveis por objetos de uma classe derivada sem afetar a correção do programa.** | **Muitas interfaces específicas são melhores do que uma única interface geral.** | **Dependências devem ser abstraídas, ou seja, dependemos de abstrações e não de implementações concretas.** |

|  |  |  |  |
| --- | --- | --- | --- |
| Factory Method | Abstract Factory | Singleton | Builder |
| Define uma interface para criar objetos, mas permite que as subclasses alterem o tipo de objetos que serão criados. | **Fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.** | **Garante que uma classe tenha apenas uma instância e fornece um ponto de acesso global para essa instância.** | **Permite a criação de objetos complexos passo a passo, separando a construção de um objeto de sua representação.** |

|  |  |  |
| --- | --- | --- |
| Prototype | Proxy | Adapter |
| Permite criar novos objetos copiando um objeto existente, ao invés de criar um novo do zero. | **Fornece um substituto para um objeto real.** | **Permite que classes com interfaces incompatíveis trabalhem juntas, convertendo a interface de uma classe para outra que o cliente espera.** |