

Padrões de Projeto de Criação

Eduardo Figueiredo

<http://www.dcc.ufmg.br/~figueiredo>

Padrões de Criação

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Factory Method

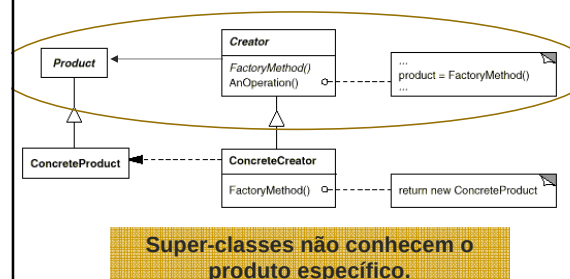
Problema (Factory Method)

- Uma classe não sabe antecipar o tipo dos objetos que a mesma precisa criar
 - É preciso adiar a instanciação de objetos para as subclasses
- Exemplo: suponha uma aplicação lida com diversos tipos de documentos
 - Ela sabe quando os documentos devem ser criados, mas não sabem que documentos criar

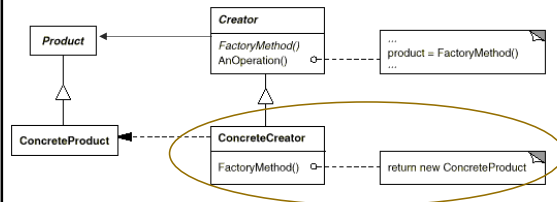
Padrão Factory Method

- Nome
 - Factory Method
- Descrição do problema
 - Slide anterior
- Descrição da solução
 - Próximo slide
- Consequências
 - Eliminam dependências de classes específicas (código lida com as interfaces)

Solução do Factory Method

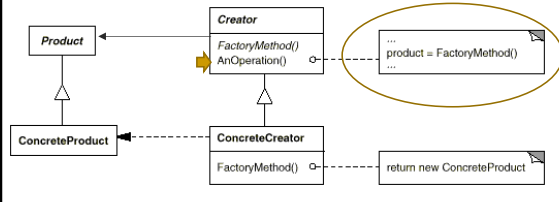


Solução do Factory Method



O método fábrica cria e retorna o objeto no momento adequado.

Solução do Factory Method



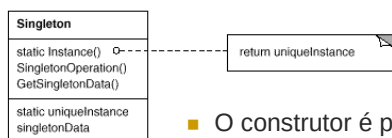
O produto geral pode seu usado pela superclasse, mesmo sem conhecer o produto específico.

Singleton

Padrão Singleton

- Nome
 - Singleton
- Descrição do problema
 - Uma classe precisa ter uma única instância
- Descrição da solução
 - Próximo slide
- Consequências
 - Fácil acesso e gerência de recursos compartilhados, como variáveis globais

Solução do Singleton



- O construtor é privado
- O método *instance()* é público e estático
 - Retorna a única instância que é guardada em uma variável de classe

Exemplo: ClasseSingleton

```
public class ClasseSingleton {
    private static ClasseSingleton instance;
    private int numInstances = 0;

    private ClasseSingleton() {
        numInstances++;
    }

    public static ClasseSingleton getInstance() {
        if (instance == null) instance = new ClasseSingleton();
        return instance;
    }

    public void printNumInstances() {
        System.out.println("numInstances = "+numInstances);
    }
}
```

Atributo que armazena a única instancia da classe.

Exemplo: ClasseSingleton

```
public class ClasseSingleton {  
    private static ClasseSingleton instance;  
    private int numInstances = 0;  
  
    private ClasseSingleton() {  
        numInstances++;  
    }  
  
    public static ClasseSingleton getInstance() {  
        if (instance == null) instance = new ClasseSingleton();  
        return instance;  
    }  
  
    public void printNumInstances() {  
        System.out.println("numInstances = "+numInstances);  
    }  
}
```

Construtor da classe é privado para evitar criação acidental de outras instancias.

Exemplo: ClasseSingleton

```
public class ClasseSingleton {  
    private static ClasseSingleton instance;  
    private int numInstances = 0;  
  
    private ClasseSingleton() {  
        numInstances++;  
    }  
  
    public static ClasseSingleton getInstance() {  
        if (instance == null) instance = new ClasseSingleton();  
        return instance;  
    }  
  
    public void printNumInstances() {  
        System.out.println("numInstances = "+numInstances);  
    }  
}
```

Única instancia só pode ser obtida pela chamada ao método *getInstance()* da classe.

Exemplo: ClasseSingletonTest

```
public class ClasseSingletonTest {  
    public static void main(String[] args) {  
        ClasseSingleton singleton = ClasseSingleton.getInstance();  
        // ClasseSingleton singleton = new ClasseSingleton();  
        singleton.printNumInstances();  
    }  
}
```

Tentar criar uma instância pela chamada ao construtor causa erro de compilação.

Bibliografia da Aula

- E. Gamma, R. Helm, R. Johnson, J. Vlissides. **Padrões de Projeto**, 1a. Edição. Bookman, 2000.
 - Padrões: Factory Method e Singleton