Modelos, Métodos e Técnicas de Engenharia de Software
Visão e análise de projeto Padrões Prática 3 – Template Method (22)

Prof. Osmar de Oliveira Braz Junior

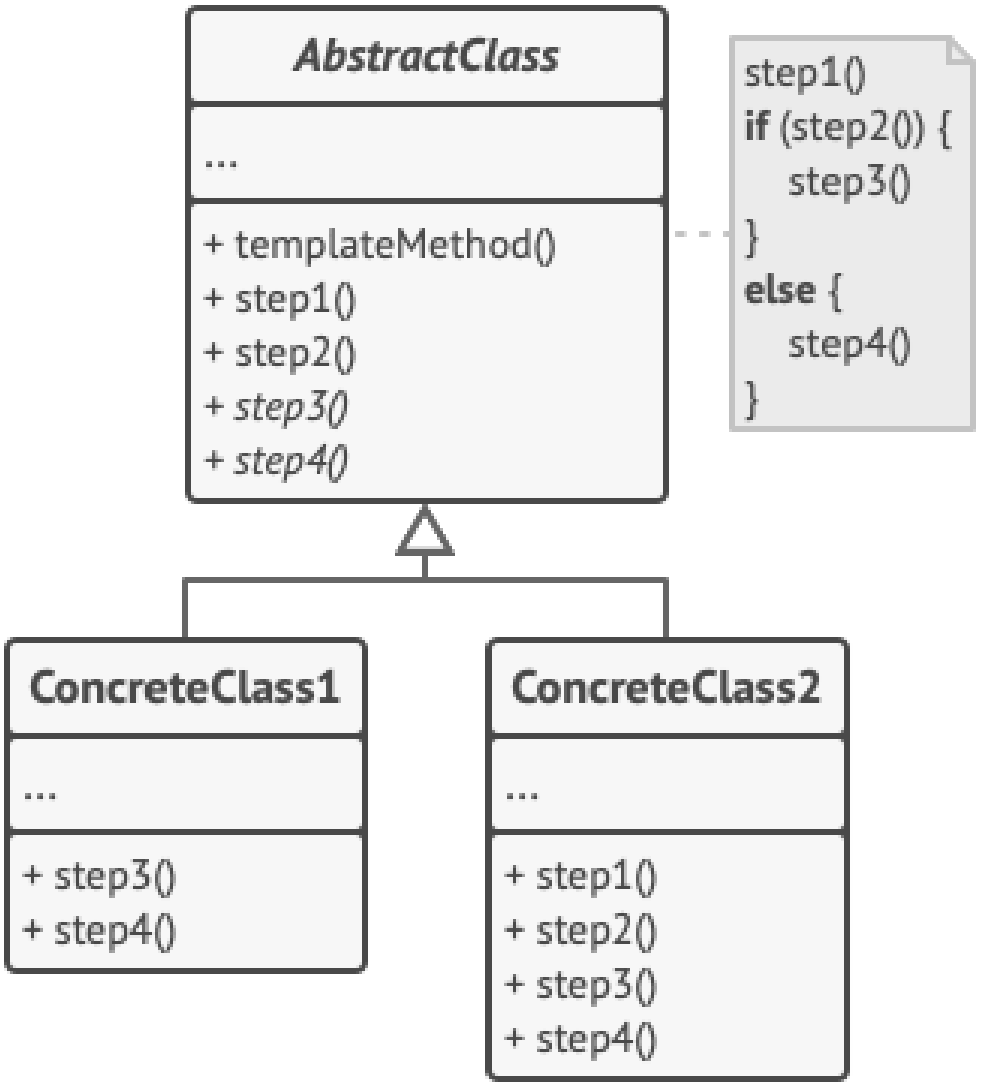
Prof. Richard Henrique de Souza

Objetivos

- Aplicar padrão comportamental ***Template Method*** em situação problema.

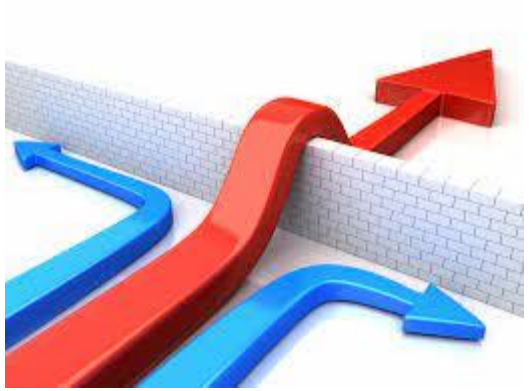
22. Template Method

Estrutura



Importante

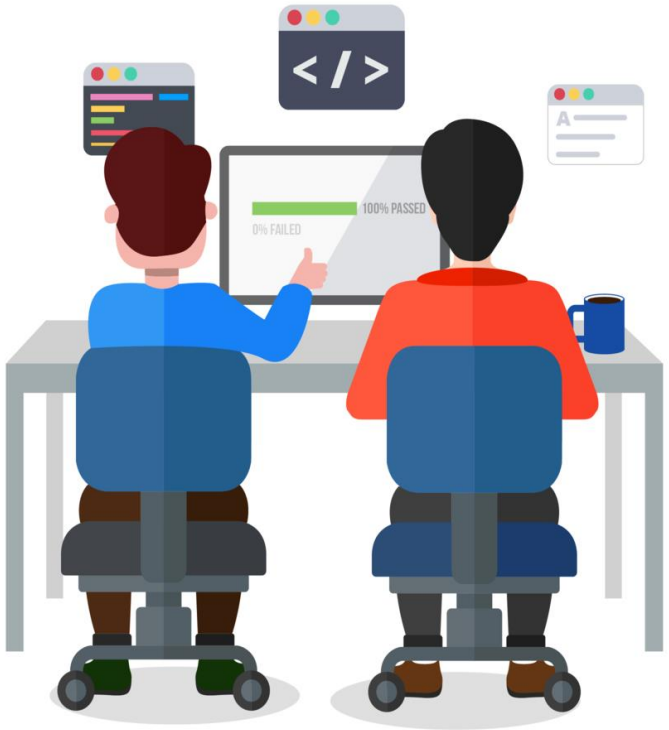
Siga os ROTEIROS !!!



Atividade em Grupo

Para esta atividade crie grupos de 2 alunos, para desenvolver a atividade segundo ***Pair Programming.***

Navegador



Piloto

Extreme Programming, metodologia ágil criada por Kent Beck em 1996.

Pair Programming

- Um é o **piloto**, responsável por escrever o código, o outro o navegador, acompanha a escrita de código e verificar se está de acordo com os **padrões do projeto** e de encontro à solução necessária.
- A intenção desta técnica é **evitar** erros de lógica, e ter um código mais confiável e melhor estruturado, utilizando-se para isso a máxima de que “**duas cabeças pensam melhor do que uma**”.

Preparação do ambiente

- Acesso a ferramenta **draw.io(https://app.diagrams.net/)** para realizar a modelagem.
- Escolha a sua linguagem de programação de preferência
- Escolha uma IDE ou o **git.dev**
- Crie um repositório no github(https://github.com/) para que todos os membros da equipe possam colaborar no desenvolvimento.

Atividade prática

1

22. Template Method
Propósito

- Definir o esqueleto de um algoritmo numa classe, delegando alguns passos às subclasses.
- Permite que as subclasses alterem partes do algoritmo, sem mudar sua estrutura geral.
- Também conhecido como: Método padrão

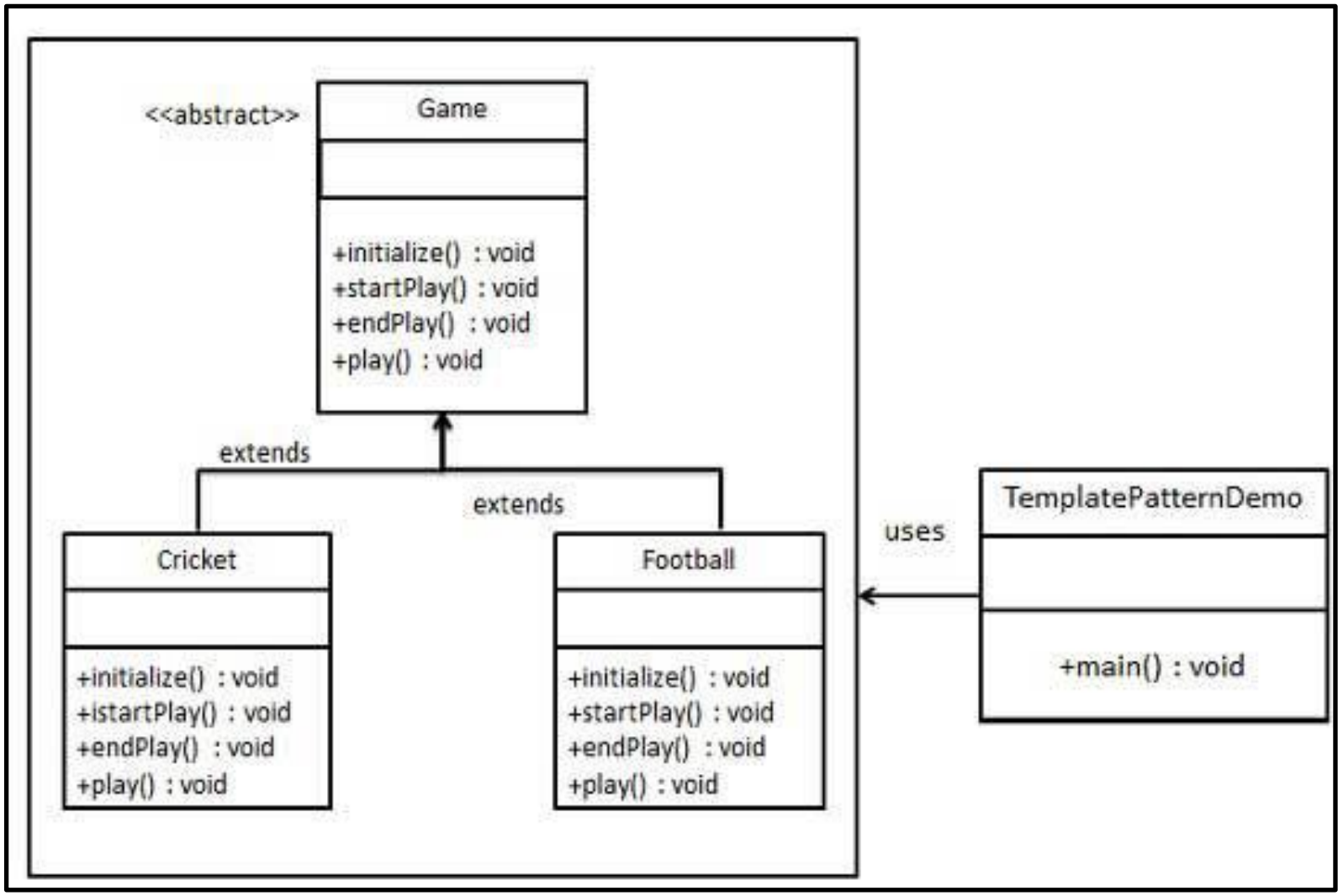
22. Template Method

- Usar este padrão quando…
  - Quiser implementar partes invariantes de um algoritmo na superclasse e deixar o restante para as subclasses;
  - Comportamento comum de subclasses deve ser generalizado para evitar duplicidade de código;
  - Quiser controlar o que as subclasses podem estender (métodos finais).

22. Template Method

- Vantagens e desvantagens
  - Reuso de código:
    - Partes de um algoritmo são reutilizadas por todas as subclasses.
  - Controle:
    - É possível permitir o que as subclasses podem estender (métodos finais).
  - Comportamento padrão extensível:
    - Superclasse pode definir o comportamento padrão e permitir sobrescrita.

22. Template Method



22. Template Method
Passo 1

Crie uma classe abstrata com um método de modelo sendo final.

Game.java

```
public abstract class Game {
   abstract void initialize();
   abstract void startPlay();
   abstract void endPlay();

   //template method
   public final void play(){

      //initialize the game
      initialize();

      //start game
      startPlay();

      //end game
      endPlay();
   }
}
```

22. Template Method
Passo 2

Crie classes concretas estendendo a classe acima.

Cricket.java

```
public class Cricket extends Game {

   @Override
   void endPlay() {
      System.out.println("Cricket Game Finished!");
   }

   @Override
   void initialize() {
      System.out.println("Cricket Game Initialized! Start playing.");
   }

   @Override
   void startPlay() {
      System.out.println("Cricket Game Started. Enjoy the game!");
   }
}
```

22. Template Method
Passo 2 - Continuação

Football.java

```java
public class Football extends Game {

    @Override
    void endPlay() {
        System.out.println("Football Game Finished!");
    }

    @Override
    void initialize() {
        System.out.println("Football Game Initialized! Start playing.");
    }

    @Override
    void startPlay() {
        System.out.println("Football Game Started. Enjoy the game!");
    }
}
```

22. Template Method
Passo 3

Use o método play() da classe Game.java para demonstrar uma forma definida de jogar o jogo.

TemplatePatternDemo.java

```
public class TemplatePatternDemo {
   public static void main(String[] args) {

      Game game = new Cricket();
      game.play();
      System.out.println();
      game = new Football();
      game.play();
   }
}
```

22. Template Method
Passo 4

- Terminamos
  - Teste sua implementação

Compile e **Mostre** o código para o professor
  - Pense, o que você fez aqui ?

Lembre de salvar no seu github

Conclusão

Os padrões comportamentais tem como principal função designar responsabilidades entre objetos.

Referências

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788580555349

- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/2613/epub/0

- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788577800476

Fim