Pontifícia Universidade Católica do Paraná
Escola Politécnica
Bacharelado em Ciência da Computação
Modelagem de Sistemas Computacionais
Prof. Edson Emilio Scalabrin
Segundo semestre de 2024
01/11/2024

Trabalho 3

Modalidade: EM GRUPO DE ATÉ 4 PESSOAS
Data de Entrega: Ver plano de Ensino

O Trabalho 3 consiste nas seguintes tarefas de modelagem e programação:

1. Implementar padrão ***Decorator***:
    - Apontamento de atividades funcionário (profissional da área de computação) .

2. Implementar padrão ***Composite***:
    - Categorização de Produtos.
    - Campi de uma Universidade.
    - Portfolio de Projetos.

3. Implementar padrão ***State***
    - Cotação de produtos.

4. Implementar padrão ***Interpretador***
    - Expressões aritméticas prefixadas.

Detalhamento dos itens do Trabalho 3.

Item 01: Aplique o padrão ***Decorator*** para modelar a seguinte aplicação:

> Os funcionários de certa empresa podem assumir qualquer uma das suas funções, inclusive de forma cumulativa. O conjunto de funções assumidas por certo funcionário é totalmente dinâmico, podendo sofrer alterações a cada dia de trabalho. Para cada uma dessas funções, há uma correspondente remuneração. Assim, o salário mensal de cada funcionário é calculado de acordo com as funções assumidas ao longo do mês.
>
> a) **No modelo fornecido** (ver arquivo “Trabalho 3 Engenharia Reversa.asta”, pacote decorator -> funcionário), as funções executadas pelos funcionários são: “Arquiteto de Software”, ‘Analista

de Sistema” e “Testador de Software”. A implementação deste modelo se encontra no código fonte “decorator.zip”.

O QUE DEVE SER FEITO?

- a) adicionar no modelo (arquivo “Trabalho 3 Engenharia Reversa.asta”, pacote decorator -> funcionário) as seguintes atividades: “Programador”, “Implantador”, “Treinamento de Usuario”.
- b) adicionar no código fonte Java (“decorator.zip”) as classes: “Programador”, “Implantador”, “Treinamento de Usuario”.
- c) adicionar na classe Teste.java (que se encontra em “decorator.zip”) dois cenários: (a) o funcionário João, realizou as seguintes atividades no mês: Programador (80h, R$ 20), Implantador (40h, R$ 40), Treinamento de Usuário (60h, 40); (b) a funcionaria Maria, realizou as seguintes atividades no mês: Analista de Sistema (90h, R$ 30), Testador de Software (20h, R$ 40), Programador (70h, R$ 20). Para codificar esses cenários, siga o exemplo que codificado na classe Teste.Java

Item 02. O padrão de projeto *Composite* permitir o tratamento de objetos individuais e composições desses objetos de maneira uniforme. Essa uniformidade diz respeito a interface de interação. A seguir são dados algumas situações de aplicação do padrão *composite*.

O QUE DEVE SER FEITO?

Deve-se implementar os modelos (diagramas de classes) descritos em: Item 02.1, Item 02.2 e Item 02.3. A implementação deve seguir o exemplo fornecido em Java, que inclui 2 pacotes de classes: “composite - > modelo” e “composite -> pgarquivo” (ver código fonte “composite.zip”). A principio todas as implementações solicitadas a seguir devem usar as classes do pacote “composite - > modelo”.

Item 02.1: Em um sistema de classificação de produtos, existem produtos e categorias, sendo que todo produto está contido em uma categoria folha e toda categoria pode conter outras categorias e categorias folhas. O diagrama de classes desta situação se encontra no arquivo “Trabalho 3 Engenharia Reversa.asta”, pacote “categoria”.

a) pede-se para implementar o referido modelo em Java; siga o exemplo de código fonte constante em “composite.zip”.

Item 02.2: Em um sistema de universitários, existem campus, blocos, andares, salas, corredores, laboratórios, auditórios, sendo que as salas, corredores, laboratórios, auditórios estão contidos em uma outra estrutura maior. O diagrama de classes desta situação se encontra no arquivo “Trabalho 3 Engenharia Reversa.asta”, pacote “universidade”.

a) pede-se para implementar o referido modelo em Java; siga o exemplo de código fonte constante em “composite.zip”.

Item 02.3: Em uma agência de projetos, os projetos são organizados em um portfolio de projetos. Cada projeto pode conter outros projetos menores. Cada projeto contem várias atividades e por fim cada atividade pode conter por sua vez várias tarefas. O diagrama de classes desta situação se encontra no arquivo "Trabalho 3 Engenharia Reversa.asta", pacote "portfolio".

a) pede-se para implementar o referido modelo em Java; siga o exemplo de código fonte constante em "composite.zip".

Item 03. O padrão de projeto *State* permitir um objeto alterar o seu comportamento em função de alterações no seu estado interno. Esse padrão torna explícitas as transições de estado.

O QUE DEVE SER FEITO?

Deve-se implementar o modelo (diagrama de classes) constante no arquivo "Trabalho 3 Engenharia Reversa.asta", pacote "State Quotacao". Caso falte algum estado para a implementação ficar completa, você deve incluir.

Sugestão de cenário de teste.

```
Pedido p1 = new Pedido();

String str = p1.solicita();    // deve retornar "Solicitado"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Solicitado"

str = p1.cotacao();    // deve retornar "Cotado"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Cotado"

str = p1.cotacao();    // deve retornar "Cotado"  Aqui não houve mudança de estado
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Cotado"

str = p1.encomenda();    // deve retornar "Encomendado"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Encomendado"

str = p1.entrega();    // deve retornar "Faturado"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Faturado"

str = p1.paga();    // deve retornar "Pago"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : Pago"

str = p1.arquiva();    // deve retornar "FIM"
System.out.println("p1-Estado : "+str)  // deve mostrar "p1-Estado : FIM"
```

- **Item 04.** O padrão de projeto *Interpreter* permitir analisar sintaticamente uma expressão.

O QUE DEVE SER FEITO?

Deve-se implementar o modelo (diagrama de classes) constante no arquivo "Trabalho 3 Engenharia Reversa.asta", pacote "interpreter". Em uma expressão aritmética, há operandos (constantes e variáveis), operadores aritméticos. Há também situações em que parte dessa expressão assume o papel de operando. Caso algum operador presente no cenário abaixo não esteja presente no modelo fornecido, você deve adicionar e implementar.

Sugestão de cenário de teste.

```
n   = 10              // constante
c1  = 20.0            // constante
c2  = 40.0            // constante
v1  = 10.0            // variável
v2  = 100.0           // variável
fat = (! n)          // fatorial de n
a   = (+ c1   v1 fat)
s   = (- (+ c1    v1)   v1)
d   = (/ (- (+ c1   v1)   v1) (+ c1   v1))
m   = (* (+ c1   v1) (- (+ c1   v1)   v1))

for (Expressao p: programa)
    System.out.println(p.toString()+ " = " + p.resolva());
```