Notas Práticas e Exercícios 2

Disciplina: Programação Orientada a Objetos

{*Anotações para uso pessoal}

1 Classes Aninhadas em Java

Classes aninhadas em Java referem-se a uma classe definida dentro de outra classe. Existem quatro tipos de classes aninhadas em Java: classes aninhadas internas, classes aninhadas estáticas, classes locais e classes anônimas. Cada tipo tem suas próprias características e propósitos.

1.1 Classes Aninhadas Internas

As classes aninhadas internas, também conhecidas como classes internas não estáticas, são definidas dentro de uma classe externa e têm acesso aos membros da classe externa, incluindo membros privados. Para instanciar uma classe interna, é necessário ter uma instância da classe externa.

A sintaxe para definir uma classe interna é a seguinte:

```
class ClasseExterna {
    // membros da classe externa

    class ClasseInterna {
        // membros da classe interna
    }
}
```

1.2 Classes Aninhadas Estáticas

As classes aninhadas estáticas, também conhecidas como classes estáticas internas, são semelhantes às classes internas, mas são definidas como estáticas. Elas não precisam de uma instância da classe externa para serem instanciadas e têm acesso apenas aos membros estáticos da classe externa.

A sintaxe para definir uma classe aninhada estática é a seguinte:

```
class ClasseExterna {
    // membros da classe externa

    static class ClasseInterna {
        // membros da classe interna
    }
}
```

1.3 Classes Locais

As classes locais são definidas dentro de um método ou bloco de código e têm acesso aos membros da classe externa e aos membros finais ou efetivamente finais declarados no escopo em que foram definidas. As classes locais são usadas principalmente para dividir um método em partes menores e mais fáceis de gerenciar.

A sintaxe para definir uma classe local é a seguinte:

```
class ClasseExterna {
    // membros da classe externa

    public void metodoExemplo() {

        class ClasseLocal {
            // membros da classe
        }

        ClasseLocal minhaClasseLocal = new ClasseLocal();
        // usar a classe local aqui

    }

}
```

1.4 Classes Anônimas

As classes anônimas são definidas como expressões e não como declarações de classe. Elas são usadas principalmente para implementar interfaces e classes abstratas em uma única linha de código. As classes anônimas não têm nome e não podem ser reutilizadas em outras partes do código.

A sintaxe para definir uma classe anônima é a seguinte:

```
InterfaceExemplo minhaInterface = new InterfaceExemplo() {
    // membros da classe anônima
};
```

2 Exercícios Práticos

2.1 Java

Implemente em Java (um de cada vez!) uma série de classes que modelem o seguinte problema: uma editora produz Livros por meio de diversos Capítulos, cada qual escrito por um autor. Para que seja feito um software orientado a objetos para essa editora armazenar e manipular seus livros será preciso implementar as classes Capítulo e Livros. Veja o diagrama abaixo. Implemente as classes em arquivos separados, e faça um programa que utilize a classe Livro incluindo um ou mais livros. As classes deverão ter as seguintes caracter ísticas: a) Capitulo possuirá os atributos: título, numero de páginas (número inteiro ¿=1) e sobrenome do autor. Deverá fornecer um construtor que inicialize todos as variáveis membro, e métodos de acesso ao título do capítulo, autor e número de páginas. Métodos modicadores não são necessários b) Livro possuirá os atributos: editor (sobrenome da pessoa que organizou o livro), título do livro, ano (número inteiro ¿= 2011), edição (número inteiro ¿ 0), preço e capítulos (um ou mais objetos Capítulo). A classe livro deverá conter um construtor para inicializar as variáveis membro, exceto os capítulos. Deverá fornecer um método para inserir objetos capítulo, e um método para imprimir as informações do livro (que deverá imprimir editor, título, ano, edição preço e uma lista de capítulos).

1. Crie uma classe `Tempo` com três atributos: horas, minutos e segundos. Crie dois construtores: um para inicializar os atributos com valor 0 e outro para inicializar os atributos com valores passados como argumentos. Crie métodos para:

    a. Funcionar como getter e setter;

    b. Imprimir os atributos no formato hh:mm:ss;

    c. Subtrair dois objetos e colocar o resultado no objeto que o chamou;

    d. Somar que soma dois objetos e colocar o resultado no objeto que o chamou;

    e. Sobrecarregue este último método para que retorne um objeto com o resultado da operação.

2. Crie uma classe `Estacionamento` para armazenar dados de um estacionamento. Os atributos devem representar a placa e modelo do carro além da hora de entrada e saída do estacionamento. Utilize dois objetos da classe Tempo criada no exercício anterior. Crie métodos para:

    a. Funcionar como getter e setter;

    b. Inicializar os dados com vazio ou zero;

    c. Imprimir os dados de um carro estacionado;

    d. Calcular e retornar o valor a ser pago pelo carro estacionado. Considere o preço de R$1,50 por hora. Utilize o método da classe Tempo.

3. Implemente em Java (um de cada vez!) uma série de classes que modelem o seguinte problema: uma editora produz Livros por meio de diversos Capítulos, cada qual escrito por um autor. Para que seja feito um software orientado a objetos para essa editora armazenar e manipular seus livros será preciso implementar as classes Capítulo e Livros. Veja o diagrama abaixo. Implemente as classes em arquivos separados, e faça um programa que utilize a classe Livro incluindo um ou mais livros. As classes deverão ter as seguintes características:

    a) Capitulo possuirá os atributos: título, número de páginas (número inteiro ¿= 1) e sobrenome do autor. Deverá fornecer um construtor que inicialize todos as variáveis membro, e métodos de acesso ao título do capítulo, autor e número de páginas. Métodos modificadores não são necessários.

    b) Livro possuirá os atributos: editor (sobrenome da pessoa que organizou o livro), título do livro, ano (número inteiro ¿= 2011), edição (número inteiro ¿ 0), preço e capítulos (um ou mais objetos Capítulo). A classe livro deverá conter um construtor para inicializar as variáveis membro, exceto os capítulos. Deverá fornecer um método para inserir objetos capítulo, e um método para imprimir as informações do livro (que deverá imprimir editor, título, ano, edição preço e uma lista de capítulos).

4. Crie uma classe `Fração`, com as seguintes características:

    a. Crie um construtor que evita o valor zero ou negativo como denominador;

    b. Sobrecarregue os operadores de adição, subtração, multiplicação e divisão desta classe;

5. Faça uma classe Vetor com um método soma que receba dois vetores de inteiros e retorne um terceiro vetor com a soma dos valores dos dois vetores passados como argumento. Sobrecarregue esse método para que também possa receber dois vetores de números decimais e retorne um terceiro vetor com a soma dos valores dos dois vetores passados como argumento.

6. Implemente uma classe Circulo com um método calculaArea que receba o raio do círculo e retorne a área. Sobrecarregue esse método para que também possa receber o diâmetro do círculo e retorne a área.

7. Crie uma classe Matriz com um método soma que receba duas matrizes de inteiros e retorne uma terceira matriz com a soma dos valores das duas matrizes passadas como argumento. Sobrecarregue esse método para que também possa receber duas matrizes de números decimais e retorne uma terceira matriz com a soma dos valores das duas matrizes passadas como argumento.

2.2 UML

Dos exercícios 1 a 4, na criação dos diagramas, em alguns casos, a decisão de qual a melhor alternativa para modelagem pode ser sua. Fique a vontade.

Crie um diagrama de classes para:

1. Um sistema de gerenciamento de tarefas com as seguintes características:
   - Tarefas com nome, descrição, data de criação e data de conclusão.
   - Atribuição de tarefas a usuários.
   - Usuários com nome e e-mail, capazes de criar, editar e concluir tarefas.
2. Um sistema de locadora de filmes com as seguintes características:
   - Filmes com título, ano de lançamento, diretor e gênero.
   - Aluguel de filmes por clientes por um período de tempo, com pagamento de taxa.
   - Criação de novos filmes e clientes, e consulta do histórico de aluguéis de um cliente.
3. Um sistema de gerenciamento de projetos com as seguintes características:
   - Projetos com nome, descrição, data de início e data de término, compostos por várias tarefas.
   - Tarefas com nome, descrição, data de criação e data de conclusão, atribuídas a membros da equipe.
   - Membros da equipe com nome e e-mail, capazes de criar, editar e concluir tarefas.
   - Criação de novos projetos, membros da equipe e atribuição de tarefas a eles.
4. Um sistema de vendas online com as seguintes características:
   - Clientes realizando pedidos de vários produtos.
   - Produtos com nome, preço e descrição.
   - Criação de novos produtos, realização de pedidos pelos clientes e cálculo do valor total da compra.
   - Consulta do histórico de pedidos de um cliente.

Analise os códigos abaixo e crie os seus diagramas de classes UML. Explique esses diagramas e escolhas.

Código 1

```java
public class Pessoa {
    private String nome;
    private int idade;
    private Endereco endereco;

    public Pessoa(String nome, int idade, Endereco endereco) {
        this.nome = nome;
        this.idade = idade;
        this.endereco = endereco;
    }
```

```
11
12      public String getNome() {
13          return nome;
14      }
15
16      public void setNome(String nome) {
17          this.nome = nome;
18      }
19
20      public int getIdade() {
21          return idade;
22      }
23
24      public void setIdade(int idade) {
25          this.idade = idade;
26      }
27
28      public Endereco getEndereco() {
29          return endereco;
30      }
31  }
32
33  public class Endereco {
34      private String rua;
35      private int numero;
36      private String cidade;
37      private String estado;
38
39      public Endereco(String rua, int numero, String cidade, String estado) {
40          this.rua = rua;
41          this.numero = numero;
42          this.cidade = cidade;
43          this.estado = estado;
44      }
45
46      public String getRua() {
47          return rua;
48      }
49
50      public void setRua(String rua) {
51          this.rua = rua;
52      }
53
54      public int getNumero() {
55          return numero;
56      }
57
58      public void setNumero(int numero) {
59          this.numero = numero;
60      }
61
```

```
62      public String getCidade() {
63          return cidade;
64      }
65
66      public void setCidade(String cidade) {
67          this.cidade = cidade;
68      }
69
70      public String getEstado() {
71          return estado;
72      }
73
74      public void setEstado(String estado) {
75          this.estado = estado;
76      }
77  }
78
79  public class Empresa {
80      private String nome;
81      private ArrayList<Funcionario> funcionarios;
82
83      public Empresa(String nome) {
84          this.nome = nome;
85          funcionarios = new ArrayList<Funcionario>();
86      }
87
88      public String getNome() {
89          return nome;
90      }
91
92      public void setNome(String nome) {
93          this.nome = nome;
94      }
95
96      public void adicionarFuncionario(Funcionario funcionario) {
97          funcionarios.add(funcionario);
98      }
99
100     public void removerFuncionario(Funcionario funcionario) {
101         funcionarios.remove(funcionario);
102     }
103
104     public ArrayList<Funcionario> getFuncionarios() {
105         return funcionarios;
106     }
107 }
108
109 public class Funcionario {
110     private String nome;
111     private int idade;
112     private Empresa empresa;
```

```
113
114     public Funcionario(String nome, int idade, Empresa empresa) {
115         this.nome = nome;
116         this.idade = idade;
117         this.empresa = empresa;
118     }
119
120     public String getNome() {
121         return nome;
122     }
123
124     public void setNome(String nome) {
125         this.nome = nome;
126     }
127
128     public int getIdade() {
129         return idade;
130     }
131
132     public void setIdade(int idade) {
133         this.idade = idade;
134     }
135
136     public Empresa getEmpresa() {
137         return empresa;
138     }
139 }
```

Código 2

```
1     public class Escola {
2     private String nome;
3     private ArrayList<Turma> turmas;
4
5     public Escola(String nome) {
6         this.nome = nome;
7         turmas = new ArrayList<Turma>();
8     }
9
10    public String getNome() {
11        return nome;
12    }
13
14    public void setNome(String nome) {
15        this.nome = nome;
16    }
17
18    public void adicionarTurma(Turma turma) {
19        turmas.add(turma);
20    }
21
22    public void removerTurma(Turma turma) {
```

```
23          turmas.remove(turma);
24      }
25
26      public ArrayList<Turma> getTurmas() {
27          return turmas;
28      }
29  }
30
31  public class Turma {
32      private int ano;
33      private String letra;
34      private ArrayList<Aluno> alunos;
35      private Professor professor;
36
37      public Turma(int ano, String letra, Professor professor) {
38          this.ano = ano;
39          this.letra = letra;
40          this.professor = professor;
41          alunos = new ArrayList<Aluno>();
42      }
43
44      public int getAno() {
45          return ano;
46      }
47
48      public void setAno(int ano) {
49          this.ano = ano;
50      }
51
52      public String getLetra() {
53          return letra;
54      }
55
56      public void setLetra(String letra) {
57          this.letra = letra;
58      }
59
60      public void adicionarAluno(Aluno aluno) {
61          alunos.add(aluno);
62      }
63
64      public void removerAluno(Aluno aluno) {
65          alunos.remove(aluno);
66      }
67
68      public ArrayList<Aluno> getAlunos() {
69          return alunos;
70      }
71
72      public Professor getProfessor() {
73          return professor;
```

```
    }
}

public class Aluno {
    private String nome;
    private int idade;
    private Turma turma;

    public Aluno(String nome, int idade, Turma turma) {
        this.nome = nome;
        this.idade = idade;
        this.turma = turma;
        turma.adicionarAluno(this);
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public Turma getTurma() {
        return turma;
    }
}

public class Professor {
    private String nome;
    private ArrayList<Turma> turmas;

    public Professor(String nome) {
        this.nome = nome;
        turmas = new ArrayList<Turma>();
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
```

```
125     }
126
127     public void adicionarTurma(Turma turma) {
128         turmas.add(turma);
129     }
130
131     public void removerTurma(Turma turma) {
132         turmas.remove(turma);
133     }
134
135     public ArrayList<Turma> getTurmas() {
136         return turmas;
137     }
138 }
```