6,8

Universidade Federal do Maranhão - UFMA
Centro de Ciências Exatas e Tecnologia
Departamento de Informática
Disciplina: Linguagem de Programação II – 2024.1
Professor: Tiago Bonini Borchartt
Nome: _____ Data: 22/04/24

Primeira Avaliação

4,8

1. Crie as classes a seguir, em Java, utilizando os conceitos e padrões da orientação a objetos (5,0):

Atleta
– nome : String
– altura : float
– peso : float
– num_camisa : int

Equipe
– nome : String
– modalidade : Esporte
– atletas : ArrayList<Atleta>
+ adicionarAtleta(Atleta a) : bool
+ removerAtleta(String nome, int num) : bool
+ mostrarEquipe( ) : void
. . .

Esporte
– nome : String
– num_integrantes : int

Onde os métodos:

- adicionarAtleta – Adiciona um novo atleta na equipe, desde que não ultrapasse o número máximo de integrantes daquela modalidade, retorna *true* se a operação foi realizar com sucesso e *false* caso contrário;
- removerAtleta – Remove o atleta que tenha o nome e o número informados, retorna *true* se a operação foi realizar com sucesso e *false* caso contrário;
- mostrarEquipe – mostra todos os dados da equipe: nome, modalidade, número de integrantes e as informações completas dos atletas.

1,0

2. Conceitue o que são classes e objetos na Programação Orientada à Objetos. Especifique em qual contexto cada conceito é utilizado. Por fim, dê exemplos práticos. (1,0)

1,0

3. Quais são as característica principais que uma linguagem de programação precisa abranger, para que seja considerada orientada à objetos? (1,0)

para uma linguagem de programação ser considerada orientada a objetos, ela deve garantir as seguintes propriedades: abstração,

22 / 04 / 24

2. Classe é a modelo que define as características e métodos que são compartilhados por uma série de objetos; o objeto é o elemento real ou abstrato que é definido por uma classe, possuindo parâmetros, métodos e atributos na programação, quando se quer criar um sistema que vai catalogar e / ou definir uma coleção de elementos que possuem características semelhantes em comum, a P.O.O resolve o problema. Um exemplo é o sistema de uma loja, de uma biblioteca, que vão definir vários objetos com características comuns a sua classe (cliente na loja, livro na biblioteca).

3. A P.O.O deve conter:
- Abstração: capacidade de representar elementos do mundo real ou abstrato em forma algorítmica.
- Encapsulamento: proteção da escrita dos dados de uma classe e seus objetos.
- Herança: capacidade de uma classe filha ter herdar características e métodos de uma classe mãe.
- Polimorfismo: capacidade de um objeto em assumir diversas formas
- Modularidade: propriedade da linguagem que permite a divisão do código em partes menores, de modo que seja possível alterá-las sem afetar o funcionamento completo do código

22 / 04 / 24

1. (Atleta.java)

```
Public class Atleta {
    private String nome;
    private float altura;
    private float pesa;
    private int num_camisa;
}

    public void Atleta(String nome, float altura, float pesa, int num_camisa) {
        this.nome = nome;
        this.altura = altura;
        this.pesa = pesa;
        this.num_camisa = num_camisa;
    }
    public void retNome(String nome) {
        this.nome = nome;
    }
    public void retAltura(float altura) {
        this.altura = altura;
    }
    public void retpesa(float pesa) {
        this.pesa = pesa;
    }
    public void retnum_camisa(int num_camisa) {
        this.num_camisa = num_camisa;
    }
    public String getNome() { return this.nome; }
    public float getAltura() { return this.altura; }
    public float getpesa() { return this.pesa; }
    public int getnum_camisa() { return this.num_camisa; }
```

22 / 04 / 24

(Esporte.java)

```
Public class Esporte {
    private String name;
    private int nmm_integrantes;
}

public void Esporte (String name, int nmm_integrantes) {
    this.name = name;
    this.nmm_integrantes = nmm_integrantes;
}
public String getName () { return this.name; }
public int getNmm_integrantes () { return this.nmm_integrantes; }
public void setName (String name) {
    this.name = name;
}
public void setNmm_integrantes (int nmm_integrantes) {
    this.nmm_integrantes = nmm_integrantes;
}
```

(Equipe.java)

```
import java.util.ArrayList;
Public class Equipe {
    private String name;
    private Esporte modalidade;
    private ArrayList<Atleta> atletas;
}

public void Equipe (String name, Esporte modalidade, ArrayList<Atleta> atletas) {
    this.name = name;
    this.modalidade = modalidade;
    this.atletas = atletas;
}

~~public String getName (String name) { return this.name; }~~
```

22 / 04 / 24

```
public String getNome() { return this.nome; }
public Esporte getModalidade() { return this.modalidade; }
public ArrayList<Atleta> getAtletas() { return this.atletas; }
public void setNome(String nome) {
    this.nome = nome;
}
public void setModalidade(Esporte modalidade) {
    this.modalidade = modalidade;
}
public void setAtletas(ArrayList<Atleta> atletas) {
    this.atletas = atletas;
}
public boolean adicionarAtleta(Atleta a) {
    if (atletas.length < modalidade.getNumIntegrantes()) {
        this.Atletas.add(a);
        return True;
    }
    return false;
}
public boolean removerAtleta(String nome, int num) {
    for(int i = 0; i < atletas.length; i++) {
        if (atletas[i].getNome() == nome && atletas[i].getNum() == num) {
            atletas.pop(atletas[i]);
            return True;
        }
    }
    return false;
}
```

22 / 04 / 24

```
public void mostrarAtletas() {
    for (int i = 0; i < atletas.size(); i++) {
        System.out.println(atletas.get(i).getNome());
        System.out.println(atletas[i].getAltura());
        System.out.println(atletas[i].getPeso());
        System.out.println(atletas[i].getIdade());
    }
    System.out.println(this.nome);
    System.out.println(this.modalidade.getNome());
    System.out.println(this.atletas.size());
}
```