RESIDÊNCIA EM TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

ALBERT SILVA DE JESUS

LINGUAGEM JAVA

Atividade PI-P001 da linguagem de programação java apresentada ao docente Alvaro Degas, como requisito parcial para aprovação no modulo introdutório.

Ilhéus – Bahia

2023

Pesquise no Google e responda às perguntas abaixo:

1. O que é uma classe em Java e qual é a diferença entre uma classe e um objeto? Dê 5 exemplos mostrando-os em C++ e em Java.

R: Pensando como uma questão de uma construtora que precisa construir casas:

Classe é a planta, é o planejamento, é o modelo a ser seguido para que a casa seja construída dentro de certas características. É algo abstrato, é algo lógico. Lá está definido todos os elementos que a casa terá e as características básicas de como eles serão compostas. Ela só existe no código. Classe tipifica o que será modelado por ela. Ela determina os estados possíveis e os comportamentos que os objetos podem ter.

O **objeto** é a casa. É algo concreto, algo físico. Nele os elementos estão de fato presentes ali. É algo palpável (em termos de computador), é algo que pode ser manipulado. Ele existe na memória, durante a execução da aplicação. Objeto possui valores para os estados definidos e chamam os comportamentos definidos executando os algoritmos. Tem um tempo de vida transitório.

Então o objeto é uma instância da classe. Na classe você pode dizer que aquele objeto terá uma cor, no objeto você diz qual é a cor, só pode dizer isso porque foi definido na classe que essa informação deve estar no objeto.

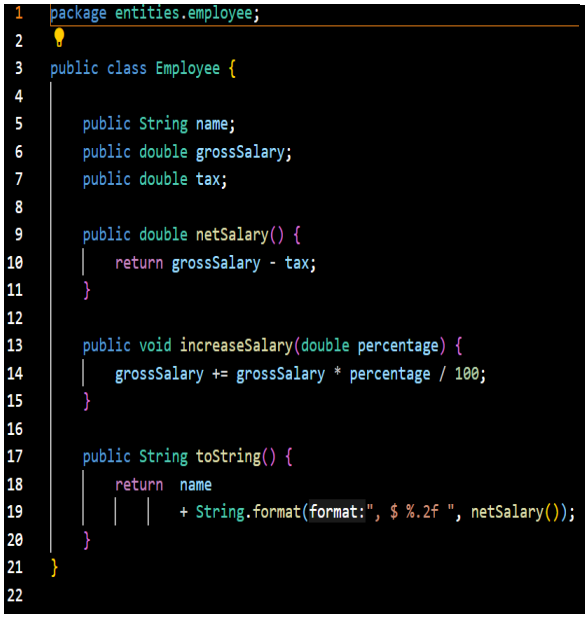
Exemplo em C++:

```cpp
#ifndef CARRINHODECOMPRAS_H
#define CARRINHODECOMPRAS_H
#include <iostream>
#include <vector>
#include "Produto.h"

using namespace std;

#pragma once

class CarrinhoDeCompras
{
private:
    Produto produto;
    vector<pair<Produto, int>> itens_no_carrinho;
    double valorTotal = 0;

    CarrinhoDeCompras() {}

    ~CarrinhoDeCompras() {}

    double getValorTotal()
    {
        return this->valorTotal;
    }

    void setValorTotal(double valorTotal)
    {
        this->valorTotal = valorTotal;
    }

    void adicionarProduto(Produto produto, double quantidade)
    {
        itens_no_carrinho.push_back(make_pair(produto, quantidade));
    }
```

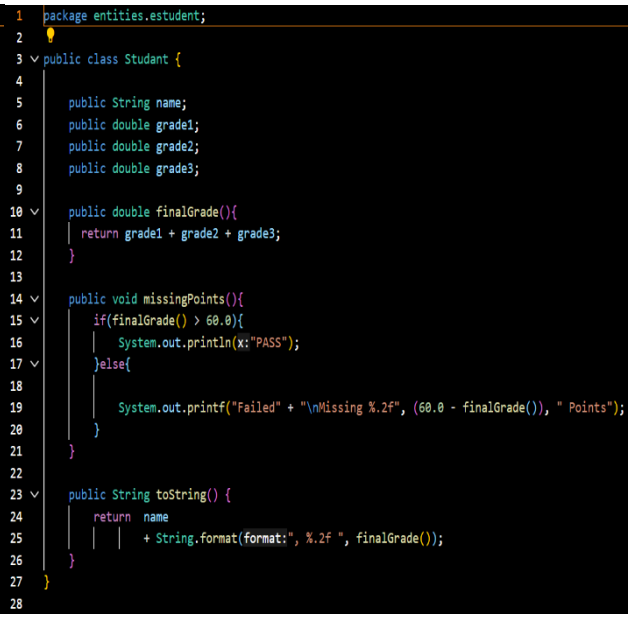
```cpp
class persistencia
{
public:

    static void salvaMausoleu(mausoleu novo)
    {
        ofstream outMausoleus;
        outMausoleus.open("mausoleus.txt", ios_base::app);
        if (outMausoleus.is_open())
        {
            outMausoleus << novo.getId() << endl;
            outMausoleus << novo.getLocalizacao() << endl;
            outMausoleus.close();
        }
    }

    static void salvaPacienteNoMausoleu(int id, paciente novo)
    {...
    }

    static void recuperaMausoleus()
    {...
    }
};

int mausoleu::contador = 1;

int main()
{
    limpaTela();
    persistencia::recuperaMausoleus();
    int op;

    do
    {
        cout << "Opcoes" << endl;
        cout << "1. Incluir Mausoleu" << endl;
        cout << "2. Listar Mausoleus" << endl;
        cout << "3. Recepcionar paciente" << endl;
        cout << "0. Sair" << endl;
        cout << "Digite opcao: ";
        cin >> op;
```

Exemplo em Java:

```java
public class Pessoa {
    // Atributos da classe
    private String nome;
    private int idade;
    // Construtor da classe
    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }
    // Método para obter o nome da pessoa
    public String getNome() {
        return nome;
    }
    // Método para definir o nome da pessoa
    public void setNome(String nome) {
        this.nome = nome;
    }
    // Método para obter a idade da pessoa
    public int getIdade() {
        return idade;
    }
    // Método para definir a idade da pessoa
    public void setIdade(int idade) {
        this.idade = idade;
    }
    // Método para exibir informações sobre a pessoa
    public void exibirInformacoes() {
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade + " anos");
    }
   public static void main(String[] args) {
        // Exemplo de uso da classe
        Pessoa pessoa1 = new Pessoa("João", 25);
        pessoa1.exibirInformacoes();
        // Alterando a idade da pessoa1
        pessoa1.setIdade(26);
        pessoa1.exibirInformacoes()
    }
}
```



```java
package entities.employee;

public class Employee {

    public String name;
    public double grossSalary;
    public double tax;

    public double netSalary() {
        return grossSalary - tax;
    }

    public void increaseSalary(double percentage) {
        grossSalary += grossSalary * percentage / 100;
    }

    public String toString() {
        return  name
                + String.format(format:", $ %.2f ", netSalary());
    }
}

```



```java
package entities.estudent;

public class Studant {

    public String name;
    public double grade1;
    public double grade2;
    public double grade3;

    public double finalGrade(){
        return grade1 + grade2 + grade3;
    }

    public void missingPoints(){
        if(finalGrade() > 60.0){
            System.out.println(x:"PASS");
        }else{

            System.out.printf("Failed" + "\nMissing %.2f", (60.0 - finalGrade()), " Points");
        }
    }

    public String toString() {
        return  name
                + String.format(format:", %.2f ", finalGrade());
    }
}

```

2. Como você declara uma variável em Java e quais são os tipos de dados primitivos mais comuns? Faça um paralelo entre isso e a mesma coisa na linguagem C++

R: Em Java, você declara uma variável especificando o tipo de dado, seguido pelo nome da variável.

Exemplo: **int** idade;

String nome;

double altura;

Tipos de dados comuns em Java:

- **int**: Representa números inteiros.
- **double**: Representa números de ponto flutuante.
- **float**: Também representa números de ponto flutuante, mas com menor precisão que o **double**.
- **boolean**: Representa valores booleanos (verdadeiro ou falso).
- **char**: Representa um único caractere.

Paralelo entre a declaração de variáveis em Java e C++:

C++	Java
int idade;	**int** idade;
std::string nome;	**string** nome;
double altura;	**double** altura;

Ambas as linguagens seguem uma abordagem semelhante na declaração de variáveis. No entanto, há algumas diferenças:

- Em Java, as **strings são objetos** e são tratadas de forma especial, enquanto em C++, você usa a **classe std::string** para representar strings.

- Em C++, a declaração de variáveis pode envolver o uso da **palavra-chave auto** para dedução de tipo, o que não é uma característica presente no Java.

3. Explique o conceito de herança em Java e como você pode criar uma subclasse a partir de uma classe existente. Faça um paralelo com C++, apresentando 5 exemplos.

R: A herança é um conceito fundamental na programação orientada a objetos que permite que uma classe (subclasse ou classe derivada) herde características e comportamentos de outra classe (superclasse ou classe base). Em Java, a herança é implementada usando a palavra-chave **extends**.

Exemplo de criar subclasse:

```
// Superclasse (classe base)
class Animal {
    void emitirSom() {
        System.out.println("Algum som genérico");
    }
}
// Subclasse (classe derivada)
class Cachorro extends Animal {
    void latir() {
        System.out.println("Au au!");
    }
}
public class TesteHeranca {
    public static void main(String[] args) {
        // Criando uma instância da subclasse
        Cachorro meuCachorro = new Cachorro();
        // Chamando método da superclasse
        meuCachorro.emitirSom();  // Resultado: Algum som genérico
        // Chamando método da subclasse
        meuCachorro.latir();  // Resultado: Au au!
    }
}
```

Paralelo entre Java e C++, exemplos:

Herança básica java:

```
class Animal {
```

```
    void emitirSom() {
        System.out.println("Algum som genérico");
    }
}
class Cachorro extends Animal {
    void latir() {
        System.out.println("Au au!");
    }
}
```

Herança básica C++:

```
class Animal {
public:
    void emitirSom() {
        std::cout << "Algum som genérico" << std::endl;
    }
};
class Cachorro : public Animal {
public:
    void latir() {
        std::cout << "Au au!" << std::endl;
    }
};
```

Herança múltipla (C++ Only)

```
class Pato {
public:
    void fazerQuack() {
        std::cout << "Quack quack!" << std::endl;
    }
};
class CachorroPato : public Cachorro, public Pato {
public:
    void nadar() {
        std::cout << "Nadando!" << std::endl;
    }
};
```

Acesso protegido:

Java:

```
class Animal {
    protected void emitirSom() {
        System.out.println("Algum som genérico");
    }
}
class Cachorro extends Animal {
    void latir() {
        emitirSom(); // Pode acessar emitirSom() porque é protegido
```

```
        System.out.println("Au au!");
    }
}
```

4. Quando declaramos uma variável em Java, temos, na verdade, um ponteiro. Em C++ é diferente. Discorra sobre esse aspecto.

R: Em Java, ao declarar uma variável de um tipo primitivo (como int, double, etc.), você está criando uma variável que armazena diretamente o valor do dado, não um ponteiro. Ou seja, a variável em si contém o valor real do dado.

Exemplo: **int** idade = 25;

Nesse exemplo, idade é uma variável que contém diretamente o valor 25. Não há ponteiros envolvidos no contexto de tipos primitivos

No entanto, quando se trata de objetos em Java, as variáveis armazenam referências aos objetos, não os objetos em si. As referências são basicamente ponteiros para a posição na memória onde o objeto está alocado.

Exemplo: **String** nome = "João";

Nesse caso, **nome** é uma variável que contém uma referência para o objeto de String "João".

Em **C++,** a linguagem oferece a capacidade de trabalhar diretamente com ponteiros. Você pode declarar um ponteiro para um tipo de dado e, em seguida, alocar ou desalocar memória dinamicamente usando operadores **new** e **delete**.

Exemplo: **int*** ponteiroIdade = new int;

*****ponteiroIdade = 25;

Neste exemplo, **ponteiroIdade** é um ponteiro para um inteiro, e a memória é alocada dinamicamente para armazenar o valor **25**. É responsabilidade do programador liberar essa memória usando **delete** quando não for mais necessária.