Unicamp
MC322
Prof. Esther Colombini

Programação Orientada a Objetos - Quiz 1

Erik Yuji Goto

RA: 234009

Campinas
2020

1. (b) int compute(int a, int b)

2. (b) O código está incorreto
(d) O atributo i de Prova está inacessível no main

3.

```java
public class numJava {
    /*Declaramos os metodos estaticos para que seja possivel
       usa-los sem precisar instanciar objetos de numJava*/
    public static int somaInt(int a, int b){
        return a + b;
    }

    public static int subInt(int a, int b){
        return a - b;
    }

    public static int somaString(String a, String b){
        return Integer.parseInt(a) + Integer.parseInt(b);
    }
    public static int subString(String a, String b){
        return Integer.parseInt(a) - Integer.parseInt(b);
    }
}
```

4. Qualquer programa em Java passa por cinco passos até que ocorra sua efetiva execução:

- Edição(do código fonte);
- Compilação(código fonte para Bytecode);
- Carregamento(carregar o .class na memória);
- Verificação(Bytecode);
- Execução(código fonte para Bytecode).

Na execução a Máquina Virtual Java(JVM) executa as instruções dos Bytecodes. Essa execução ocorre combinando interpretação e compilação Just-in-time(JIT).

5.
(a) Seu valor, uma vez inicializado, não pode ser modificado. **(F)**
(b) São declaradas com a palavra-chave static. **(V)**
(c) Pode ser alterada dentro de um método de instância. **(V)**
(d) Existe somente uma cópia de cada variável, independente do número de objetos. **(V)**

6. (c) A classe não contém nenhum construtor declarado

7.

```java
private Classe varClasse;
```

Nesta linha declaramos apenas a classe **varClasse** do tipo **Classe**. É como um "aviso"para o sistema de que existe uma classe do tipo **Classe** e podemos instanciá-la a qualquer momento.

```
1     private Classe varClasse = new Classe() ;
```

Já neste trecho estamos declarando uma Classe e ao mesmo tempo *instanciando* a mesma. Dessa maneira, já podemos acessar as variáveis e métodos da classe. Não é necessário chamar o construtor posteriormente

8. Atributos de Instância: Estes podem ser de qualquer tipo(int, boolean, String, etc) e também podem ser objetos de outra classe. Comumente, são inicializados no início do código e declarados da seguinte forma;

```
1     <visibilidade> tipo Nome;
```

Atributos de Classe: Assim como as variáveis de instância podem ser de qualquer tipo básico. Entretanto, esse tipo de variável tem uma característica única: elas são criadas uma única vez para a classe como um todo, e não para cada objeto. Portanto, as variáveis de classe existem assim que a classe é carregada na memória.

São úteis para contar a quantidade de objetos da classe criados, ou para calcular o valor médio de uma determinada propriedade. E declaradas da seguinte forma;

```
1     <visibilidade> static tipo Nome;
```

Variáveis Locais: São variáveis que só podem ser acessadas localmente. Só podem ser usadas dentro do algoritmo que foram declaradas.

Parâmetros: Os parâmetros são tipos de variáveis que auxiliam no funcionamento de métodos. São variáveis temporárias que serão utilizadas dentro das funções que estas são chamadas. EX:

```
1     public void NomeClasse(int parametro1, String parametro2,
          boolean parametro3){
2         if (parametro3 == true){
3             x = parametro1;
4             System.out.println(parametro2);
5         }
6     }
```

9. Public: Um elemento definido como *public* pode ser acessado por qualquer objeto de qualquer classe. É a classe que não restringe nenhum acesso.

Private: Já elementos com *private* só podem ser acessados por outros elemtos dentro da classe em que o elemento private foi declarado. É usado para para atributos e métodos internos da classe.

Protected: A visibilidade *protected* é o meio termo entre as duas outras. Ela só pode ser acessada por **classes** que **herdam** da classe onde o *protected* foi declarado.

Segundo o *princípio do menor privilégio:* a ideia é **atribuir privilégios** a determinados elementos apenas quando necessário para executar determinada tarefa. Portanto, a visibilidade contribui para limitar os acessos que cada classe pode ter, ou conceder. Essa

funcionalidade é muito útil para o conceito de *encapsulamento* na programação orientada a objetos.

O encapsulamento permite tornar os códigos flexíveis e reutilizáveis para novas implementações; combinado com o princípio do menor privilégio podemos controlar o acesso a atributos e métodos.

10. 1001

11.

(a) cada elemento do array é uma referência para outro array
(b) representam um estrutura de dados com pelo menos duas dimensões
(c) pode ser utilizado para representar valores armazenados em uma tabela organizada em linhas e colunas
(d) para um array bidimensional, as linhas não precisam ter o mesmo tamanho

12. (b) O código está incorreto
(c) Prova11 é um tipo de Prova1
(d) O atributo i está inacessível em Prova11