Fatec Faculdade de Tecnologia	FACULDADE DE TECNOLOGIA- FATEC Curso: Desenvolvimento de Software Multiplataforma Disciplina: Técnicas de Programação II

Aluno:Kevin de Almeida Brandão	RA:1091392323045	Data: 03/04/2024

1.

```
Digite 10 números inteiros:
1
2
3
4
5
6
5
4
3
23
Digite um valor de referência inteiro:
4
Números maiores que o valor de referência:
5 6 5 23
```

2.

```
etailsInExceptionMessages -cp "C:\Users\Kevin Brandao\
Digite o código do produto nº 0 :
2
Digite o valor unitário do produto nº 0 :
22
Digite a quantidade vendida nº 0 :
22
Digite o código do produto nº 1 :
1
Digite o valor unitário do produto nº 1 :
2
Digite a quantidade vendida nº 1 :
```

3.

```
Digite os 10 valores para o vetor w:
1
2
3
4
5
6
7
8
9
12
Digite os 10 valores para o vetor k:
2
33
4
5
66
7
8
9
7
5
O resultado do cálculo é: 203
```

4.

```
Vetor A:
6 7 50 40 34 0 18 44 21 32 39 25 26 18 31 44 4 35 18 10
Vetor B:
36 5 36 35 26 33 21 23 10 13 24 26 32 29 38 16 12 16 14 18
--------------------------------
Vetor C (A+B ordenado e sem repetição):
0 4 5 6 7 10 12 13 14 16 18 21 23 24 25 26 29 31 32 33 34 35 36 38 39 40 44 50
```

5.

```
Digite um número para geração da tabuada:
3
Tabuada do 3:

Soma:
3 + 0 = 3
3 + 1 = 4
3 + 2 = 5
3 + 3 = 6
3 + 4 = 7
3 + 5 = 8
3 + 6 = 9
3 + 7 = 10
3 + 8 = 11
3 + 9 = 12
3 + 10 = 13

Multiplicação:
3 x 0 = 0
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

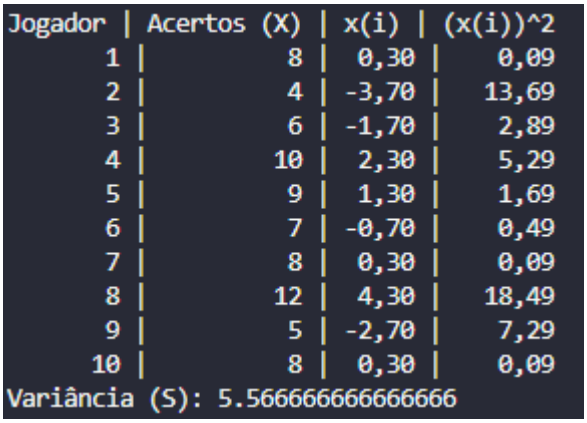
6.

```
Valor de t: 1.9766034691407826
Vetor W:
81 5 78 5 29 73 61 54 44 13 77 25 15 15 13 75 42 15 73 19 90 67 38 7 84 65 48 18 86 40 58 5 68 94 3 57 97 36 0 72 86 50 30 53 72 85 15 82 31 56 76 3 41 13 14 58 48 98 80 37 3 89 3 25 73 30 9 90 35 83 39 4 21 24 50 31 68
19 69 94 71 16 10 12 85 41 4 92 85 8 77 74 91 54 78 3 32 69 74 65
Vetor J:
25 98 68 95 10 98 71 50 22 44 26 51 71 47 19 42 81 47 56 18 62 22 77 41 84 16 83 78 24 54 67 87 19 81 87 52 36 76 11 54 57 87 48 27 28 99 11 40 19 51 92 11 10 44 61 22 77 2 79 11 60 1 33 51 34 58 96 50 13 45 86 18 51 93
88 98 29 69 37 52 47 59 97 49 47 93 72 98 39 84 41 10 15 75 9 16 65 13 92 34
```

7.

```
Desvio padrão: 29.685876498706563
```

8.



Jogador	Acertos (X)	x(i)	(x(i))^2
1	8	0,30	0,09
2	4	-3,70	13,69
3	6	-1,70	2,89
4	10	2,30	5,29
5	9	1,30	1,69
6	7	-0,70	0,49
7	8	0,30	0,09
8	12	4,30	18,49
9	5	-2,70	7,29
10	8	0,30	0,09

Variância (S): 5.566666666666666

Questões de Orientação a Objetos

1. Detalhe o significado das variáveis static e descreva algumas diferenças com as variáveis de instância.

R: Em Java, a palavra-chave *static* é usada para indicar que um campo ou método pertence à classe em si, em vez de instâncias da classe. Isso significa que:

· O valor de uma variável *static* é o mesmo para todas as instâncias da classe.

· Você pode acessar um campo *static* sem criar uma instância da classe.

· Métodos *static* podem ser chamados sem uma instância da classe e só podem acessar diretamente outros membros *static*

	Variáveis estáticas	Variáveis de instância
Memória	são armazenadas na área de memória estática. Há uma única cópia dessa variável que é compartilhada entre todas as instâncias da classe.	têm sua própria cópia na memória para cada instância da classe. Elas são armazenadas no heap e cada objeto tem sua própria cópia de variáveis de instância
Acesso	podem ser acessadas diretamente com o nome da classe (por exemplo, ClassName.staticVariable) e não requerem uma instância da classe para acessá-las.	só podem ser acessadas através de um objeto da classe (por exemplo, instance.instanceVariable) e cada objeto terá valores diferentes (ou o mesmo valor

		inicializado) para essas variáveis.
Ciclo	são inicializadas quando a classe é carregada pela primeira vez pela JVM (Java Virtual Machine), o que geralmente ocorre na primeira vez que a classe é referenciada em algum lugar do código.	são inicializadas cada vez que um novo objeto da classe é criado e morrem quando o objeto é coletado pelo Garbage Collector.
Uso	são frequentemente usadas para definir constantes ou para gerenciar o estado que deve ser compartilhado entre todas as instâncias da classe.	são usadas para definir propriedades que são específicas para cada objeto e devem ter valores independentes entre os objetos da classe.

2.Explique os conceitos de “abstração”, “encapsulamento” e “instância”. Enumere pelo menos 4 tipos de dados primitivos em Java.

R: Abstração:

- A abstração é um dos conceitos fundamentais da programação orientada a objetos (POO) e refere-se à simplificação de um sistema complexo, focando nos aspectos mais relevantes para um determinado contexto. Em programação, a abstração permite definir classes, interfaces e métodos que representam entidades do mundo real de forma simplificada e manipulável pelo código.

Encapsulamento:

- O encapsulamento é um princípio de POO que combina os dados e os métodos que operam sobre esses dados em uma única unidade chamada classe. Ele restringe o acesso direto aos dados de uma classe, encapsulando-os e permitindo o acesso apenas por meio de métodos públicos. Isso promove a segurança, ocultação e modularidade do código.

Instância:

- Em programação orientada a objetos, uma instância é um objeto específico criado a partir de uma classe. Uma classe é como um modelo ou uma planta baixa que define a estrutura e o comportamento dos objetos que serão criados a partir dela. Ao criar uma instância de uma classe, estamos criando um objeto que possui seus próprios dados e pode executar os métodos definidos na classe.

Tipos de dados primitivos em Java:

- Java possui tipos de dados primitivos que representam valores simples e básicos. Aqui estão quatro deles:
    - int: Representa números inteiros, como 1, -5, 100, etc.
    - double: Representa números decimais de ponto flutuante, como 3.14, -0.5, 100.0, etc.
    - boolean: Representa valores lógicos verdadeiro ou falso.
    - char: Representa um único caractere Unicode, como 'a', 'B', '&', etc.

3. Dada a classe abaixo:

a) As variáveis de classe na classe Blue são:

static int color;
public static String NAME = "Blue";

b) O construtor para a classe Blue pode ser identificado pela sua declaração que tem o mesmo nome da classe e não tem tipo de retorno:

public Blue(int hue) { ... }

c) Para implementar o método setHue sem mudar o código fornecido, você precisará adicionar uma variável de instância à classe para armazenar o valor de hue. O método setHue irá atualizar o valor desta variável:

d) Para implementar o construtor sem mudar o código escrito, você precisa inicializar a variável de instância hue dentro do construtor. Visto que o corpo do construtor não está completo na declaração fornecida, a implementação completa do construtor seria algo assim:

4. Dado o código abaixo:

```
1. public class Bird {
2.    protected static int referenceCount = 0;
3.    int a;
4.    protected void fly() { System.out.print("Flap Flap:" + a); }
5.    static int getRefCount() { return referenceCount; }
6. }
7.
8. class Nightingale extends Bird {
9.    Nightingale() { referenceCount++; }
10.
11.   public static void main(String args[]) {
12.       System.out.print("Before: " + getRefCount());
13.       Nightingale florence = new Nightingale();
14.       System.out.print("   After: " + getRefCount());
15.       florence.fly();
16.   }
17.}
```

a) Será impresso:

Before: 0 After: 1Flap Flap:0
Na linha 12, o método estático *getRefCount()* é chamado antes da criação de qualquer objeto *Nightingale*. Como *referenceCount* é uma variável estática na classe Bird e não foi modificada, seu valor inicial é 0.
Na linha 13, um novo objeto da classe *Nightingale* é criado, o que invoca o construtor *Nightingale*() e incrementa *referenceCount* em 1.
Na linha 14, o método estático *getRefCount*() é chamado novamente, agora o valor de referenceCount é 1 após o incremento pelo construtor *Nightingale*.
Na linha 15, o método *fly*() do objeto *florence* é chamado. Como a variável de instância a nunca foi inicializada, ela tem o valor padrão 0 para o tipo primitivo *int*, então "Flap Flap:0" é impresso.

b) Se forem criados 10 objetos da classe Bird e 5 da classe Nightingale, haverá:
Apenas 1 instância da variável *referenceCount*. Como *referenceCount* é uma variável estática, ela pertence à classe e é compartilhada por todas as instâncias da classe Bird e suas subclasses.
Não importa quantos objetos sejam criados; apenas uma única cópia da variável estática existe.
Serão criadas 15 instâncias da variável a. Cada novo objeto de Bird ou *Nightingale* terá sua própria cópia da variável de instância a. Como são criados 10 objetos Bird e 5 *Nightingale*, haverá 15 instâncias de a.
Variáveis de instância são específicas de cada objeto criado a partir de uma classe e cada objeto tem sua própria cópia. Variáveis estáticas são compartilhadas entre todos os objetos da classe e existem independentemente de quaisquer objetos serem criados.

5. De acordo com o código-fonte abaixo, faça a implementação das classes “Pessoa”, “Trabalhador” e “Estudante”. A classe “Trabalhador” tem como atributos nome e salário. A classe “Estudante” tem como atributos nome e idade:

```
abstract class Pessoa {
        protected String nome;

        public Pessoa(String nome) {
        this.nome = nome;
        }

        // metodo abstrato reportar
        public abstract void reportar();
}
class Trabalhador extends Pessoa {
        private double salario;
```

```
        public Trabalhador(String nome, double salario) {
        super(nome);
        this.salario = salario;
        }

        //  método reportar para Trabalhador
        @Override
        public void reportar() {
        System.out.printf("Nome: %s\t Salário: %.2f\n", nome, salario);
        }
}

class Estudante extends Pessoa {
        private int idade;

        public Estudante(String nome, int idade) {
        super(nome);
        this.idade = idade;
        }

        //  método reportar para Estudante
        @Override
        public void reportar() {
        System.out.printf("Nome: %s\t Idade: %d\n", nome, idade);
        }
}
public class Main {
        public static void main(String[] params) {
        ArrayList<Pessoa> lista = new ArrayList<Pessoa>();
        Trabalhador t1 = new Trabalhador("jorge", 2000.0);
        Trabalhador t2 = new Trabalhador("jose", 5600.0);
        Estudante e1 = new Estudante("luiz", 23);
        Estudante e2 = new Estudante("tatiane", 21);

        lista.add(t1);
        lista.add(t2);
        lista.add(e1);
        lista.add(e2);

        for (Pessoa p : lista) {
        p.reportar();
        }
```

```
        }
}
```

6.

```
package questoesorientacaoobjetos;

public class Main6 {
    Run | Debug
    public static void main(String[] args) {
        // Criar uma instância de CorporateClient
        CorporateClient corporateClient = new CorporateClient(name:"Empresa XYZ", address:"Endereço XYZ", contactName:"João Contato", creditHistory:"Bom",
                creditLimit:"50000");

        // Exibir o histórico de crédito do cliente corporativo
        System.out.println("Histórico de Crédito do Cliente Corporativo: " + corporateClient.creditHistory());

        // Processar a fatura mensal do cliente corporativo
        corporateClient.monthBill(month:3); // Supondo que o 3 representa março

        IndividualClient individualClient = new IndividualClient(name:"Maria", address:"Endereço de Maria", creditCardNumber:"1234-5678-9012-3456");

        System.out.println("Histórico de Crédito do Cliente Individual: " + individualClient.creditHistory());

        // Criar uma instância de Order
        Order order = new Order();

        // Simular o recebimento, envio e fechamento de um pedido
        order.receive();
        order.send();
        order.close();

        // Criar uma instância de OrderString
        OrderString orderString = new OrderString();
        orderString.receive(); // Esta chamada é herdada da classe Order
    }
}
```