# Prova Java

## Cabeçalho



**Curso:** Ciência da Computação **Período:** 3/5º **Turno:** Diurno

**Disciplina:** Paradigmas de Linguagens de Programação **Professor:** Dr. Ausberto S. Castro V.

**Atividade:** Prova sobre Orientação a Objetos **Data:** 19~25/11/2021

**Aluno:** João Vítor Fernandes Dias **Matrícula:** 00119110377

## Questões

### Questão 1 1 ponto 24/11/21 – 23h00 –10h11 - 25/11/21

#### Descrição

Responda detalhadamente as seguintes questões (**exemplifique**):

##### Qual é a diferença entre uma classe e um objeto?

1. Uma classe é a definição de “uma coisa”, de “um algo”, que tem algumas características (atributos), alguns desses podem ter um valor inicial padrão definidos logo em sua criação (definidos pelo construtor) e as ações feitas por ou sobre elas (procedimentos). Podemos entender que a classe é o molde que informa quais características (atributos) e/ou ações (procedimentos) esta certa coisa pode ter e fazer (ou ao invés de fazer, sofrer certa ação).
   1. Exemplo 1: Uma pessoa (classe) tem características (atributos) como cor dos olhos e data de nascimento que são informações padrões desde o nascimento (definidos pelo construtor) e também tem altura e peso que variam ao longo dos anos. Além disso, uma pessoa também é capaz de fazer certas ações (procedimentos) como piscar os olhos e se movimentar.
   2. Exemplo 2: Uma janela (classe) tem características (atributos) como material da qual é constituída e cor, sendo ambos definidos a partir de sua confecção (definidos pelo construtor), mesmo que a cor tenha sido definida inicialmente isso não a impede de ser alterada ao longo do tempo. Além disso, uma janela permite que certas ações (procedimentos) sejam feitas sobre ela como abrir, fechar e pintar.
2. O objeto é a “coisa”, o “algo”, que a classe define e que foi criado baseada nela. A partir do uso do molde (Classe), pode-se obter uma “coisa”, um “algo”, que tivesse uma listagem de características (atributos) e/ou ações (procedimentos). Embora a Classe não consiga executar nada do que define, o objeto sim. Ele se torna utilizável a partir do momento que é criado (instanciado) pela classe. “Um objeto tem identidade, estado e comportamento. Um objeto é simplesmente uma entidade tangível que apresenta um comportamento bem definido (Booch)” G. Booch et. al. Object Oriented Design with Applications. Pearson Education, 2007.
   1. Exemplo 1: A partir da classe “pessoa”, podemos criar o objeto “João” que tem características próprias (atributos) como cor dos olhos (preto) e data de nascimento (05/06/2000) que são informações padrões desde o nascimento (definidos pelo construtor) e também tem altura (1,69 m) e peso (69 kg) que variam ao longo dos anos. Além disso, João também é capaz de fazer certas ações (procedimentos) como piscar os olhos e se movimentar.
   2. Exemplo 2: A partir da classe “janela”, podemos criar o objeto “janela do quarto” que tem características próprias (atributos) como material da qual é constituída (madeira) e cor (branca), sendo ambos definidos a partir de sua confecção (definidos pelo construtor), mesmo que a cor tenha sido definida inicialmente isso não a impede de ser alterada ao longo do tempo. Além disso, a “janela do quarto” permite que certas ações (procedimentos) sejam feitas sobre ela como abrir, fechar e pintar que foram descritas pela classe “janela”.
3. Sendo assim, a diferença entre classe e objeto é que a classe é o conceito que define um conjunto de características e ações de um objeto. E um objeto é algo que possui um conjunto de características e ações conceituadas pela classe.

##### Explique como se define uma classe em Java

Uma das formas é a seguinte:

class Nome\_Da\_Classe {

*tipo\_do\_atributo* nome\_do\_atributo;

  Nome\_Da\_Classe () { //Construtor

    this.nome\_do\_atributo = valor\_inicial\_do\_atributo;

  }

*tipo\_do\_retorno\_do\_procedimento* nome\_do\_procedimento() {

instruções\_executadas\_no\_procedimento

  }

}

##### Por que uma classe é uma abstração lógica?

Porque para se criar a classe, precisamos entender de que forma algo funciona em sua essência para que então a possamos definir de forma geral. Essa definição geral é a abstração do objeto estudado da qual buscamos definir suas características e ações base. Esta abstração precisa ser coerente e lógica. Por isso uma classe é uma abstração lógica.

##### Como os objetos são criados? 25/11/21 XXhXX 00h04

Objetos são criados ao se instanciar uma classe, ou seja, partindo de uma classe que já foi abstraída e definida, podemos criar objetos derivados dessa classe. Cada objeto criado a partir de uma classe é chamado de uma instância dessa classe.

Uma das formas de se criar um objeto em Java é a seguinte:

    Nome\_da\_classe nome\_do\_objeto = new Nome\_Da\_Classe();

##### O que é coleta de lixo e como ela funciona? 25/11/21 08h22 10h11

A coleta de lixo (garbage collection) é uma forma utilizada por algumas linguagens de programação para limpeza de memória. Ela atua buscando objetos não utilizados (que são considerados lixo) e os remove da memória para limpá-la.

“Na primeira etapa, os objetos não referenciados são identificados e marcados como prontos para o garbage collector. Na segunda etapa, os objetos marcados são excluídos. Opcionalmente, a memória pode ser compactada após o garbage collector excluir objetos, portanto, os objetos restantes estão em um bloco contíguo no início do heap. O processo de compactação torna mais fácil alocar memória para novos objetos sequencialmente após o bloco de memória alocada para objetos existentes.” [Fonte: Blog Mandic](https://blog.mandic.com.br/artigos/java-garbage-collection-melhores-praticas-tutoriais-e-muito-mais/).

### Questão 2 1 ponto 25/11/21 10h15 – 16h10

#### Descrição

Desenhar todas as classes, subclasses e superclasses associadas aos seguintes objetos, indicando em cada caso, pelo menos 2 atributos e 2 métodos, e a relação de herança entre classes

1. Aparelho eletrônico
2. Casa
3. Livro
4. Mamífero
5. Planta

#### Resposta

##### Aparelho eletrônico

| **Objeto:** | **aparelhoEletronico** | **Subclasse** | **Classe** | **Superclasse** |
| --- | --- | --- | --- | --- |
| **Nome** | | Radio | SerNaoVivo | ConjutoDeAtomos |
| **Atributo** | | estacao | dataDeOrigem | quantidadeDeAtomos |
| volume | nome | tiposDeAtomos |
| **Métodos** | | sintonizar() | destruir() | mostrarEstruturaAtomica3d() |
| aumentarVolume() | manipular() | listarMoleculas() |
| **Relação de herança entre classes** | | Especializacao | Classificacao | Generalizacao |
|  |

##### Casa

| **Objeto:** | **casa** | **Subclasse** | **Classe** | **Superclasse** |
| --- | --- | --- | --- | --- |
| **Nome** | | Iglu | Moradia | Construcao |
| **Atributo** | | quantidadeDeGelo | quantidadeDeComodos | quantidadeDeTijolos |
| nomeDoConstrutor | endereco | porcentagemDeConclusao |
| **Métodos** | | derreter() | desligarLuzes() | reformar() |
| entrar() | pagarIPTU() | vender() |
| **Relação de herança entre classes** | | Especializacao | Classificacao | Generalizacao |
|  |

##### Livro

| **Objeto:** | **livro** | **Subclasse** | **Classe** | **Superclasse** |
| --- | --- | --- | --- | --- |
| **Nome** | | LivroDeFiccao | ConjuntoDePaginas | Folha |
| **Atributo** | | temaDeFiccao | quantidadeDePaginas | material |
| paginaMarcada | corDasPaginas | dimensoes |
| **Métodos** | | marcarPagina() | removerPagina() | dobrar() |
| ler() | descartar() | cortar() |
| **Relação de herança entre classes** | | Especializacao | Classificacao | Generalizacao |
|  |

##### Mamífero

| **Objeto:** | **mamifero** | **Subclasse** | **Classe** | **Superclasse** |
| --- | --- | --- | --- | --- |
| **Nome** | | Cachorro | AnimalVertebrado | Animal |
| **Atributo** | | corDoPelo | quantidadeDeVertebras | expectativaDeVida |
| nomeDaRaca | composicaoDasVertebras | dataDeNascimento |
| **Métodos** | | pegarGraveto() | fazerBarulho() | reproduzir() |
| lamber() | mover() | alimentar() |
| **Relação de herança entre classes** | | Especializacao | Classificacao | Generalizacao |
|  |

##### Planta

| **Objeto:** | **planta** | **Subclasse** | **Classe** | **Superclasse** |
| --- | --- | --- | --- | --- |
| **Nome** | | PlantaBaixa | DesenhoTecnico | Desenho |
| **Atributo** | | quantidadeDeCotas | escala | cores |
| numeroDeComodos | coordenadas | tipoDoPapel |
| **Métodos** | | corrigir() | desenhar() | rabiscar() |
| imprimir() | humanizar() | apagar() |
| **Relação de herança entre classes** | | Especializacao | Classificacao | Generalizacao |
|  |

### Questão 3 1 ponto 16h10 – 16h45

#### Descrição

Dar três exemplos de cada operação abstrata em orientação a objetos

#### Resposta

|  | **Conceito** | **Exemplo 1** | **Exemplo 2** | **Exemplo 3** |
| --- | --- | --- | --- | --- |
| **Instanciação** | **Classe -> Objeto** | Pessoa -> João | Disciplina -> Sociologia | Faculdade -> UENF |
| **Classificação** | **Objeto -> Classe** | grama -> Planta | cachorro -> Animal | brigadeiro -> Doce |
| **Especialização** | **Classe Geral -> Classe Específica** | Funcionário -> Funcionário Público | Galinha -> Galinha Caipira | Brincadeira -> Brincadeira de criança |
| **Generalização** | **Classe Específica -> Classe Geral** | Planta -> Ser Vivo | Folclore Brasileiro -> Folclore | Doença de pele -> Doença |
| **Decomposição** | **Objeto Complexo -> Objeto Particionado** | Samsumg -> Tela | Monitor -> Painel LCD | Cadeira -> Forro |
| **Agregação** | **Objeto Particionado -> Objeto Complexo** | Autofalante -> Rádio | Paus -> Canoa | Tijolo -> Construção |

### Questão 4 1 ponto 16h45 – XX

#### Descrição

Escreva um programa em Java que execute as quatro operações aritméticas de números reais

#### Resposta

class Calculadora {

*void* soma(*double* *x*, *double* *y*) {

    System.out.println(x+y);

  }

*void* subtrai(*double* *x*, *double* *y*) {

    System.out.println(x-y);

  }

*void* multiplica(*double* *x*, *double* *y*) {

    System.out.println(x\*y);

  }

*void* divide(*double* *x*, *double* *y*) {

    System.out.println(x/y);

  }

}

class Main {

  public static *void* main(*String*[] *args*) {

*Calculadora* calculadoraJAVA = new Calculadora();

    calculadoraJAVA.soma(1.1,2.2);

    calculadoraJAVA.subtrai(2.2,3.3);

    calculadoraJAVA.multiplica(3.3,4.4);

    calculadoraJAVA.divide(4.4,5.5);

  }

}

### Questão 5 1 ponto XX – 17h15

#### Descrição

Escreva um programa JAVA que calcule a média de quatro notas. O programa deve mostrar o valor da média e indicar se foi aprovado ou reprovado

#### Resposta

class CalculoDeMedia {

*double* mediaAceitavel;

  CalculoDeMedia(*double* *mediaInserida*) {

    mediaAceitavel = *mediaInserida*;

  }

*void* calcularMedia(*double* *n1*, *double* *n2*, *double* *n3*, *double* *n4*) {

*double* valorMedia = (*n1*+*n2*+*n3*+*n4*)/4;

    this.aprovacao(valorMedia);

  }

*void* aprovacao(*double* *valorMediaCalculado*) {

    String msg = "";

    if (*valorMediaCalculado*>=mediaAceitavel) {

      msg = "a";

    } else {

      msg = "re";

    }

    System.out.println("O aluno está " + msg + "provado, porque sua média foi: " + *valorMediaCalculado*);

  }

}

class Main {

  public static *void* main(String[] *args*) {

    CalculoDeMedia calcularMediaJAVA = new CalculoDeMedia(6.0);

    calcularMediaJAVA.calcularMedia(1.1,2.2,8.8,9.9);

    calcularMediaJAVA.calcularMedia(2.2,3.3,8.8,9.9);

  }

}

### Questão 6 1 ponto 17h15 – 17h33

#### Descrição

Escreva um programa JAVA que calcule a soma dos inteiros maiores ou iguais a um número inteiro A e menores ou iguais a um inteiro B.

#### Resposta

class SomarInteiros {

*void* intervalo(*int* *A*, *int* *B*) {

*int* soma = 0;

    for (; *A* <= *B*; soma += *A*, *A*++);

    System.out.println(soma);

  }

}

class Main {

  public static *void* main(String[] *args*) {

    SomarInteiros somaDeIntervalo = new SomarInteiros();

    somaDeIntervalo.intervalo(1,3);

    somaDeIntervalo.intervalo(9,11);

  }

}

### Questão 7 2 pontos 18h08 – 21h06

#### Descrição

Para cada caso da questão 2, escrever um programa JAVA, mostrando as classes, as operações (métodos), a criação de objetos e outras características da Programação Orientada a Objetos.

#### Resposta

##### AparelhoEletronico

class ConjuntoDeAtomos {

*int* quantidadeDeAtomos;

  String tiposDeAtomos;

*void* mostrarEstruturaAtomica3d() {

    String mensagem1 = "Mostrando estrutura atômica 3D";

    System.out.println(mensagem1);

  }

*void* listarMoleculas() {

    String mensagem2 = "Listando moléculas";

    System.out.println(mensagem2);

  }

}

class SerNaoVivo extends ConjuntoDeAtomos{

  String dataDeOrigem;

  String nome;

*void* destruir() {

    String mensagem3 = "Destruindo";

    System.out.println(mensagem3);

  }

*void* manipular() {

    String mensagem4 = "Manipulando";

    System.out.println(mensagem4);

  }

}

class Radio extends SerNaoVivo{

*int* estacao;

*int* volume;

*void* sintonizar() {

    String mensagem5 = "Sintonizando";

    System.out.println(mensagem5);

  }

*void* aumentarVolume() {

    String mensagem6 = "Aumentando volume";

    System.out.println(mensagem6);

  }

}

class AparelhoEletronico {

  public static *void* main(String[] *args*) {

    Radio nomeDoObjeto = new Radio();

    nomeDoObjeto.mostrarEstruturaAtomica3d();

    nomeDoObjeto.listarMoleculas();

    nomeDoObjeto.destruir();

    nomeDoObjeto.manipular();

    nomeDoObjeto.sintonizar();

    nomeDoObjeto.aumentarVolume();

  }

}

##### Casa

class Construcao {

*int*  quantidadeDeTijolos;

*int*  porcentagemDeConclusao;

*void* reformar() {

    String mensagem1 = "Reformando";

    System.out.println(mensagem1);

  }

*void* vender() {

    String mensagem2 = "Vendendo";

    System.out.println(mensagem2);

  }

}

class Moradia extends Construcao {

*int*     quantidadeDeComodos;

  String  endereco;

*void* desligarLuzes() {

    String mensagem3 = "Desligando Luzes";

    System.out.println(mensagem3);

  }

*void* pagarIPTU() {

    String mensagem4 = "pagar IPTU";

    System.out.println(mensagem4);

  }

}

class Iglu extends Moradia {

*int*     quantidadeDeGelo;

  String  nomeDoConstrutor;

*void* derreter() {

    String mensagem5 = "Derretendo";

    System.out.println(mensagem5);

  }

*void* entrar() {

    String mensagem6 = "Entrando";

    System.out.println(mensagem6);

  }

}

class Casa {

  public static *void* main(String[] *args*) {

    Iglu nome\_do\_objeto = new Iglu();

    nome\_do\_objeto.reformar();

    nome\_do\_objeto.vender();

    nome\_do\_objeto.desligarLuzes();

    nome\_do\_objeto.pagarIPTU();

    nome\_do\_objeto.derreter();

    nome\_do\_objeto.entrar();

  }

}

##### Livro

class Folha {

  String  material;

  String  dimensoes;

*void* dobrar() {

    String mensagem1 = "Dobrando";

    System.out.println(mensagem1);

  }

*void* cortar() {

    String mensagem2 = "Cortando";

    System.out.println(mensagem2);

  }

}

class ConjuntoDePaginas extends Folha {

*int*     quantidadeDePaginas;

  String  corDasPaginas;

*void* removerPagina() {

    String mensagem3 = "Romovendo pagina";

    System.out.println(mensagem3);

  }

*void* descartar() {

    String mensagem4 = "Descartando";

    System.out.println(mensagem4);

  }

}

class LivroDeFiccao extends ConjuntoDePaginas {

  String temaDeFiccao;

*int*    paginaMarcada;

*void* marcarPagina() {

    String mensagem5 = "Marcando Pagina";

    System.out.println(mensagem5);

  }

*void* ler() {

    String mensagem6 = "Lendo";

    System.out.println(mensagem6);

  }

}

class Livro {

  public static *void* main(String[] *args*) {

    LivroDeFiccao duna = new LivroDeFiccao();

    duna.dobrar();

    duna.cortar();

    duna.removerPagina();

    duna.descartar();

    duna.marcarPagina();

    duna.ler();

  }

}

##### Mamifero

class Animal {

  String  expectativaDeVida;

  String  dataDeNascimento;

*void* reproduzir() {

    String mensagem1 = "Reproduzindo";

    System.out.println(mensagem1);

  }

*void* alimentar() {

    String mensagem2 = "Alimentando";

    System.out.println(mensagem2);

  }

}

class AnimalVertebrado extends Animal {

*int*     quantidadeDeVertebras;

  String  composicaoDasVertebras;

*void* fazerBarulho() {

    String mensagem3 = "Fazendo Barulho";

    System.out.println(mensagem3);

  }

*void* mover() {

    String mensagem4 = "Movendo";

    System.out.println(mensagem4);

  }

}

class Cachorro extends AnimalVertebrado {

  String  corDoPelo;

  String  nomeDaRaca;

*void* pegarGraveto() {

    String mensagem5 = "Pegando Graveto";

    System.out.println(mensagem5);

  }

*void* lamber() {

    String mensagem6 = "Lambendo";

    System.out.println(mensagem6);

  }

}

class Mamifero {

  public static *void* main(String[] *args*) {

    Cachorro poodle = new Cachorro();

    poodle.reproduzir();

    poodle.alimentar();

    poodle.fazerBarulho();

    poodle.mover();

    poodle.pegarGraveto();

    poodle.lamber();

  }

}

##### Planta

class Desenho {

  String  cores;

  String  tipoDoPapel;

*void* rabiscar() {

    String mensagem1 = "Rabiscando";

    System.out.println(mensagem1);

  }

*void* apagar() {

    String mensagem2 = "Apagando";

    System.out.println(mensagem2);

  }

}

class DesenhoTecnico extends Desenho {

*int*     escala;

  String  coordenadas;

*void* desenhar() {

    String mensagem3 = "Desenhando";

    System.out.println(mensagem3);

  }

*void* humanizar() {

    String mensagem4 = "Humanizando";

    System.out.println(mensagem4);

  }

}

class PlantaBaixa extends DesenhoTecnico {

*int*  quantidadeDeCotas;

*int*  numeroDeComodos;

*void* corrigir() {

    String mensagem5 = "Corrigindo";

    System.out.println(mensagem5);

  }

*void* imprimir() {

    String mensagem6 = "Imprimindo";

    System.out.println(mensagem6);

  }

}

class Planta {

  public static *void* main(String[] *args*) {

    PlantaBaixa plantaBaixaEletrica = new PlantaBaixa();

    plantaBaixaEletrica.rabiscar();

    plantaBaixaEletrica.apagar();

    plantaBaixaEletrica.desenhar();

    plantaBaixaEletrica.humanizar();

    plantaBaixaEletrica.corrigir();

    plantaBaixaEletrica.imprimir();

  }

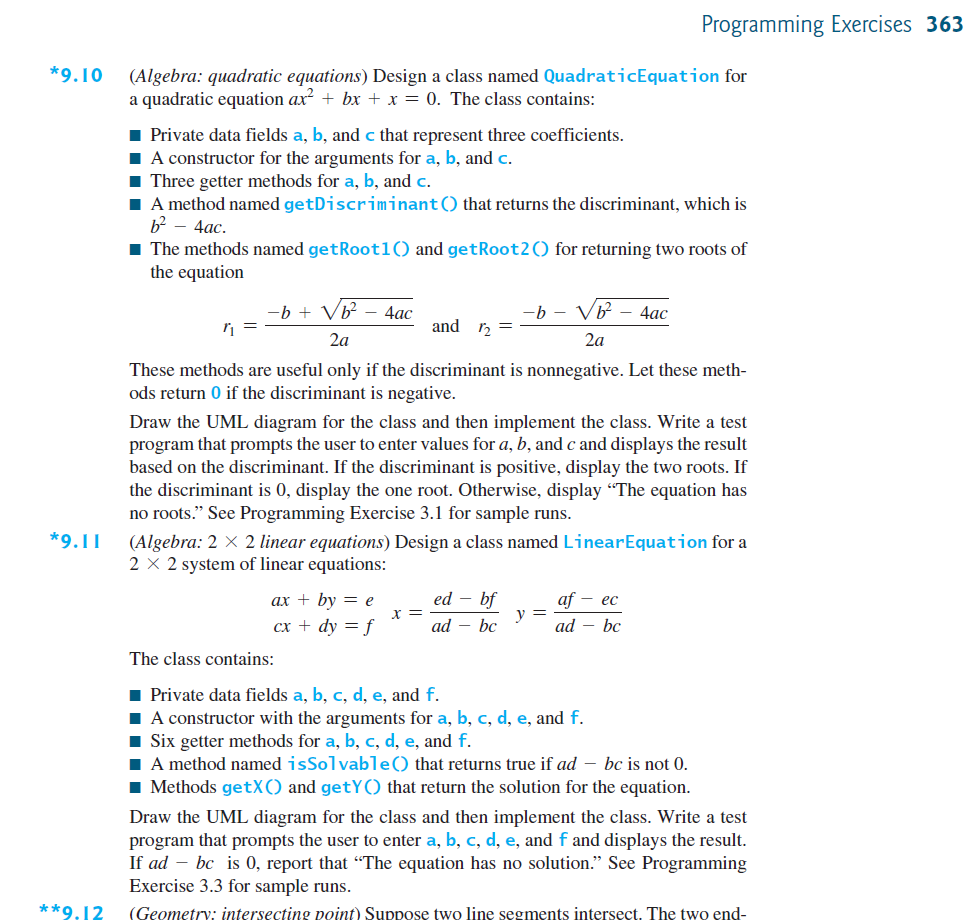
}

### Questão 8 2 pontos 22h00 – 23h03

#### Descrição

Resolver a Questão **9.10** e **9.11** (pág.363) do livro “Introduction to JAVA Programming” , Y. Daniel Liang, 2014. Não precisa desenhar os diagramas UML. “Implemente a classe” = “Faça um programa em Java”

#### Imagem



#### 9.10 (Algebra: quadratic equations)

##### Descrição

Design a class named QuadraticEquation for a quadratic equation ax² + bx + x = 0. The class contains:

* Private data fields a, b, and c that represent three coefficients.
* A constructor for the arguments for a, b, and c.
* Three getter methods for a, b, and c.
* A method named getDiscriminant() that returns the discriminant, which is b² - 4ac.
* The methods named getRoot() and getRoot2 () for returning two roots of the equation

r1 = (-b + √(b²-4ac))/(2a) and r2 = (-b - √(b²-4ac))/(2a)

These methods are useful only if the discriminant is nonnegative. Let these methods return 0 if the discriminant is negative.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter values for a, b, and c and displays the result based on the discriminant. If the discriminant is positive, display the two roots. If the discriminant is 0, display the one root. Otherwise, display “The equation has no roots.” See Programming Exercise 3.1 for sample runs.

##### Código

import java.util.Scanner;

class QuadraticEquation {

  private *float* a;

  private *float* b;

  private *float* c;

  QuadraticEquation(*float* *constructA*, *float* *constructB*, *float* *constructC*) {

    this.a = *constructA*;

    this.b = *constructB*;

    this.c = *constructC*;

  }

*float* getA(){

    return a;

  }

*float* getB(){

    return b;

  }

*float* getC(){

    return c;

  }

*float* getDiscriminant() {

    return b\*b-4\*a\*c;

  }

*float* getRoot() {

    if(getDiscriminant()<0) {

      return 0;

    }

    return (-b + (*float*)Math.sqrt(b\*b-4\*a\*c))/(2\*a);

  }

*float* getRoot2() {

    if(getDiscriminant()<0) {

      return 0;

    }

    return (-b - (*float*)Math.sqrt(b\*b-4\*a\*c))/(2\*a);

  }

}

class Q9\_10 {

  public static *void* main(String[] *args*) {

    Scanner input = new Scanner(System.in);

    System.out.print("A:");

*float* a = input.nextFloat();

    System.out.print("B:");

*float* b = input.nextFloat();

    System.out.print("C:");

*float* c = input.nextFloat();

    QuadraticEquation calculadoraQuadratica = new QuadraticEquation(a, b, c);

*float* r1 = calculadoraQuadratica.getRoot();

*float* r2 = calculadoraQuadratica.getRoot2();

*float* discriminant = calculadoraQuadratica.getDiscriminant();

    if (discriminant > 0) {

      System.out.println("r1="+r1+"; r2="+r2+";");

    } else if (discriminant == 0) {

      System.out.println("r1="+r1+";");

    } else {

      System.out.println("The equation has no roots.");

    }

    input.close();

  }

}

#### 9.11 (Algebra: 2 X 2 linear equations)

##### Descrição

Design a class named LinearEquation for a

2 X 2 system of linear equations:

ax+by=e x=(ed-bf)/(ad-bc)

cx+dy=f y=(af-ec)/(ad-bc)

The class contains:

* Private data fields a, b, c, d, e, and f.
* A constructor with the arguments for a, b, c, d, e, and f.
* Six getter methods for a, b, c, d, e, and f.
* A method named isSolvable() that returns true if ad-bc is not 0.
* Methods getX() and getY() that return the solution for the equation.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter a, b, c, d, e, and f and displays the result. If ad — bc is 0, report that “The equation has no solution.” See Programming Exercise 3.3 for sample runs.

##### Código

import java.util.Scanner;

class LinearEquation {

  private *float* a;

  private *float* b;

  private *float* c;

  private *float* d;

  private *float* e;

  private *float* f;

  LinearEquation(*float* *constructA*, *float* *constructB*, *float* *constructC*, *float* *constructD*, *float* *constructE*, *float* *constructF*) {

    this.a = *constructA*;

    this.b = *constructB*;

    this.c = *constructC*;

    this.d = *constructD*;

    this.e = *constructE*;

    this.f = *constructF*;

  }

*float* getA(){

    return a;

  }

*float* getB(){

    return b;

  }

*float* getC(){

    return c;

  }

*float* getD(){

    return d;

  }

*float* getE(){

    return e;

  }

*float* getF(){

    return f;

  }

*boolean* isSolvable() {

    return (a\*d-b\*c != 0);

  }

*float* getX() {

    return (e\*d-b\*f)/(a\*d-b\*c);

  }

*float* getY() {

    return (a\*f-e\*c)/(a\*d-b\*c);

  }

}

class Q9\_11 {

  public static *void* main(String[] *args*) {

    Scanner input = new Scanner(System.in);

    System.out.print("a:");

*float* a = input.nextFloat();

    System.out.print("b:");

*float* b = input.nextFloat();

    System.out.print("c:");

*float* c = input.nextFloat();

    System.out.print("d:");

*float* d = input.nextFloat();

    System.out.print("e:");

*float* e = input.nextFloat();

    System.out.print("f:");

*float* f = input.nextFloat();

    LinearEquation calculadoraLinear = new LinearEquation(a, b, c, d, e, f);

*float* X = calculadoraLinear.getX();

*float* Y = calculadoraLinear.getY();

*boolean* solvable = calculadoraLinear.isSolvable();

    System.out.println("Eq1: ("+a+")\*X + ("+b+")\*Y = ("+e+");");

    System.out.println("Eq2: ("+c+")\*X + ("+d+")\*Y = ("+f+");");

    if (!solvable) {

      System.out.println("The equation has no solution.");

    } else {

      System.out.println("X="+X+"; Y="+Y+";");

    }

    input.close();

  }

}