

Programação Orientada a Objetos

JAVA

Caderno de Exercícios

Programação Orientada a Objetos

Lista 1

- 1) O Java foi criado a partir de quais linguagens de programação?

- 2) Quais as principais razões que levaram os engenheiros da Sun a desenvolver uma nova linguagem de programação?

- 3) Por que um sistema escrito em Java pode rodar em qualquer plataforma?

- 4) Marque com "X" na alternativa correta. Para rodar uma aplicação Java, por mais simples que seja, é necessário possuir uma Java Virtual Machine.
() Verdadeiro () Falso

- 5) Qual o principal papel do JCP (Java Community Process)?

Anotações

2

Programação Orientada a Objetos

6) Por quem é formado o JCP?

7) Marque “V” para verdadeiro ou “F” para falso.

- a. () A Oracle, como dona da tecnologia poderá mudar os rumos do Java a qualquer tempo.
- b. () Para a Oracle fazer uma alteração no Java será necessário se submeter ao JCP.
- c. () O JCP é o responsável por defender os interesses da indústria da comunidade Java e da Oracle.

8) Como está estruturada a plataforma Java?

9) O que é Java?

Anotações

3

Programação Orientada a Objetos

10) Marque com um "X" na alternativa correta.

É possível compilar um código Java para uma plataforma específica?

() Verdadeiro () Falso

11) Quais as principais características do Java?

12) Qual a função do Garbage Collector?

13) Quais são as três tecnologias Java para desenvolvimento de aplicativos?

Anotações

4

Programação Orientada a Objetos

14) O Java é compilado ou interpretado?

15) Para que serve a Java Virtual Machine?

1) Quais são as convenções estabelecidas para a declaração de classes, métodos e variáveis?

2) Como inserimos comentários em um código Java?

3) Um dos principais motivos que contribuíram para o desenvolvimento da linguagem Java foi:

- a. ☐ O nome da linguagem.
- b. ☐ O desenvolvimento da Internet.
- c. ☐ A linguagem ser relativamente simples.
- d. ☐ O desempenho da linguagem em termos de velocidade.

4) Por que o aspecto da utilização do Java em multiplataforma é muito importante para os programadores?

Anotações

5

Programação Orientada a Objetos

- 5) Qual das características seguintes não diz respeito a linguagem Java:
- a. ☐ Pode ser executada em qualquer computador, independente de existir uma máquina virtual java instalada.
 - b. ☐ É uma linguagem compilada e interpretada.
 - c. ☐ O desempenho dos aplicativos escritos em Java, com relação à velocidade de execução, é inferior à maioria das linguagens de programação.
 - d. ☐ É uma linguagem com um bom nível de segurança.
- 6) A sequência de desenvolvimento de um programa em Java é:
- a. ☐ Compilação, digitação e execução.
 - b. ☐ Digitação, execução e compilação.
 - c. ☐ Digitação, compilação e execução.
 - d. ☐ Digitação, execução e testes de funcionamento.
- 7) Qual a principal característica que distingue a plataforma Java das demais existentes?
- 8) Para a linguagem Java, as variáveis PATH e CLASSPATH correspondem a:
- a. ☐ Variáveis usadas em um programa Java.
 - b. ☐ Uma variável de ambiente e um caminho para a execução dos programas Java.
 - c. ☐ Um caminho para encontrar as classes e um caminho para encontrar os aplicativos da linguagem Java.
 - d. ☐ Um caminho para encontrar os aplicativos e um caminho para encontrar as classes da linguagem Java.
- 9) Qual a diferença entre uma variável do tipo primitivo e uma variável do tipo reference?
- 10) Quais são os tipos primitivos da linguagem Java?

Anotações

6

Programação Orientada a Objetos

Lista 2

1) O que são variáveis locais?

2) Dado o código a seguir:

```
public class App1{
    public static void main(String args[ ]){
        String s1 = args[1];
        String s2 = args[2];
        String s3 = args[3];
        String s4 = args[4];
        System.out.println("args[2] = " + s2);
    }
}
```

e a chamada de linha de comando sendo

java App1 1 2 3 4

Qual o resultado? (Selecione um)

- a. args[2] = 2;
- b. args[2] = 3;
- c. args[2] = null;
- d. A compilação falhará;
- e. Uma exceção será lançada no tempo de execução.

3) Dado o código a seguir,

```
public class Foo {
    public void main( String[] args ) {
        System.out.println( "Hello" + args[0] );
    }
}
```

e a chamada de linha de comando sendo:

Anotações

Programação Orientada a Objetos

java Foo world

Qual o resultado? (Selecione um)

- a. Hello
- b. Hello Foo
- c. Hello world
- d. A compilação falhará;
- e. O código não executará.

4) Dado os códigos abaixo, analise a sintaxe e assinale (V) para verdadeiro (F) para falso.

- a. ☐ int x = 10.45;
- b. ☐ float f = 3.4;
- c. ☐ char a = "S";
- d. ☐ boolean z = true; if(z){ }

Anotações

8

Programação Orientada a Objetos

Lista 3

- 1) Crie um programa que recebe três nomes quaisquer por meio da linha de execução do programa, e os imprima na tela da seguinte maneira: o primeiro e o último nome serão impressos na primeira linha um após o outro, o outro nome (segundo) será impresso na segunda linha.

- 2) Faça um programa que receba a quantidade e o valor de três produtos, no seguinte formato: quantidade1 valor1 quantidade2 valor2 quantidade3 valor3. O programa deve calcular esses valores seguindo a fórmula $\text{total} = \text{quantidade1} \times \text{valor1} + \text{quantidade2} \times \text{valor2} + \text{quantidade3} \times \text{valor3}$. O valor total deve ser apresentado no final da execução.

- 3) Crie um programa que receba a largura e o comprimento de um lote de terra e mostre a área total existente.

- 4) Crie um programa que receba quatro valores quaisquer e mostre a média, somatório entre eles e o resto da divisão do somatório por cada um dos valores.

Anotações

9

Programação Orientada a Objetos

Lista 4

- 1) Faça um aplicativo que receba três valores inteiros na linha de comando e mostre o maior dentre eles.
- 2) Faça um programa que apresente o total da soma dos cem primeiros números inteiros ($1+2+3+\dots+99+100$).
- 3) Faça um aplicativo que calcule o produto dos inteiros ímpares de 1 a 15 e exiba o resultado na tela.
- 4) Crie uma classe que gere um numero aleatório entre os valores máximo e mínimo recebidos do usuário na linha de comando.

Anotações

10

Programação Orientada a Objetos

Lista 5

- 1) O que é um array?

- 2) Como um array unidimensional pode ser declarado?

- 3) Todo array declarado como uma variável local deve ser inicializado.

() Verdadeiro () Falso

- 4) Todo array é uma variável do tipo:

() Primitivo () Reference

- 5) Como obter o tamanho de um array?

- 6) Como um array Bidimensional pode ser declarado?

Anotações

11

Programação Orientada a Objetos

Lista 6

- 1) Qual a função da classe String?

- 2) Qual método da classe String retorna o tamanho da String?

- 3) Qual método da classe String converte qualquer tipo de dados em String?

Anotações

12

Programação Orientada a Objetos

Lista 7

1) Crie um aplicativo que receba uma frase e mostre-a de forma invertida.

2) Crie um aplicativo que mostre o efeito abaixo:

```
J
Ja
Jav
Java
Jav
Ja
J
```

3) Crie uma classe que leia um parâmetro passado na linha de comando no seguinte formato: dd/mm/aaaa. Desta maneira, a classe devera ser executada como `java Exe04 11/09/2001`. A saída gerada por essa execução deve ser a impressão separada do dia, do mês e do ano - utilizando apenas os métodos da classe String.

4) Uma empresa quer transmitir dados por telefone, mas está preocupada com a possibilidade de seus telefones estarem grampeados. Todos seus dados são transmitidos como inteiros de quatro dígitos. Eles pedem para você escrever um programa que criptografará seus dados de modo que estes possam ser transmitidos mais seguramente. Seu aplicativo deve ler um inteiro de quatro dígitos inserido pelo usuário na linha de comando e criptografá-lo como segue: substitua cada dígito por (a soma deste dígito mais 1). Então troque o primeiro dígito pelo terceiro e troque o segundo pelo quarto. A seguir imprima o inteiro criptografado. Escreva um aplicativo separado que recebe como entrada um inteiro de quatro dígitos criptografado e o descriptografa para formar o número original.

5) O fatorial de um número inteiro não negativo n é escrito como $n!$ (pronuncia-se fatorial de n) e é definido como segue:

$$n! = n.(n-1) . (n-2).1 \text{ (para valores de } n \text{ maiores que ou iguais a } 1)$$

e

$$n! = 1 \text{ (para } n=0)$$

Por exemplo: $5! = 5.4.3.2.1$, o que dá 120.

Anotações

Programação Orientada a Objetos

6) Escreva um aplicativo que lê um inteiro não negativo via linha de comando, computa e imprima seu fatorial.

7) Escreva um aplicativo que recebe entradas de texto e envia o texto para saída com letras em maiúsculas e em minúsculas.

8) A série de Fibonacci 0,1,1,2,3,5,8,13,21.....

inicia com 0 e 1 e tem a prioridade de que cada número de Fibonacci subsequente é a soma dos dois anteriores que o procedem.

Escreva um aplicativo que recebe a entrada do número de vezes que deve ocorrer a série.

9) Faça um aplicativo que verifique se uma palavra é um palíndromo. Ex: Ana.

10) Uma escola precisa de um programa que controle a média das notas dos alunos de cada classe e a média das notas de todos os alunos da escola. Sabendo que essa escola possui 3 classes com 5 alunos em cada classe, gerando um total de 15 notas, crie um programa que receba as notas de cada aluno de cada classe e no final apresente a média de cada classe e a média da escola em geral.

11) Dado um vetor $v = \{91, 10, 50, 89, 45, 80, 2, 45, 3, 105, 95, 13, 26, 49, 50\}$, criar um programa a que leia um número e verifique e imprima na tela se este número existe no vetor.

12) Dado um vetor $v = \{56, 48, 9, 48, 60, 13, 24, 27, 13, 56, 85, 9, 48, 55, 9\}$, criar um programa que leia um número e informe na tela quantas vezes este número foi encontrado no vetor.

13) Criar um programa que leia um vetor de inteiros de 10 posições e efetue a soma somente dos elementos ímpares.

14) Criar um programa que leia dois vetores de inteiros de 10 posições, efetue a soma dos valores dos elementos de mesmo índice dos dois vetores colocando o resultado em um terceiro vetor. Exiba na tela o vetor resultante.

15) Criar um programa que leia um vetor de inteiros de 20 posições e mostre a quantidade de números que são múltiplos de 2.

Anotações

14

Programação Orientada a Objetos

16) Dado um vetor $v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, criar um programa que imprima este conjunto acompanhado do seu elemento simétrico em relação a sua posição no conjunto, ou seja, a impressão será: 1 - 10, 2 - 9, 3 - 8, 4 - 7, 5 - 6, 6 - 5, 7 - 4, 8 - 3, 9 - 2, 10 - 1.

17) Dado um vetor $v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, criar um programa que imprima estes valores na tela.

18) Dado um vetor $v = \{5, 10, 8, 4, 9, 16, 28, 40, 80, 10\}$ criar um programa que efetua a soma dos valores e imprima o resultado.

19) Dado um vetor $v = \{85.0, 105.0, 40.0, 90.0, 35.0, 65.0, 33.0, 22.0, 19.0, 50.0\}$, criar um programa que efetua a média dos valores e imprima o resultado.

Anotações

15

Programação Orientada a Objetos

Lista 8

Leia o que está sendo solicitado e implemente os códigos:

Exercício 1

Passo 1:

- Criar a classe Agencia
- Adicione os seguintes atributos na classe Agencia:
nrAgencia(String)
codBanco(int)

Passo 2:

- Criar a classe TestaAgencia
- Crie um objeto da classe Agencia
- Inicialize todos os atributos deste objeto.
- Imprima os valores dos atributos da classe Agencia de forma a obter o seguinte resultado:
* -----
* AGENCIA : 1
* BANCO : 234
* -----

Exercício 2

Passo 1:

- Criar a classe Cliente
- Adicione os seguintes atributos na classe Agencia:
nomeCliente(String)
cpfCliente(String)

Passo 2:

- Criar a classe TestaCliente
- Crie um objeto da classe Cliente
- Inicialize todos os atributos deste objeto.
- Imprima os valores dos atributos da classe Cliente de forma a obter o seguinte resultado:
* -----
* NOME :FULANO
* CPF : 234232323
* -----

Exercício 3

Passo 1:

- Criar a classe Conta

Anotações

16

Programação Orientada a Objetos

- Adicione os seguintes atributos na classe Conta
 - saldo(double)
 - nrAgencia(String)
 - titular(String)
 - nrConta(String)
 - codBanco(int)

Passo 2:

- Criar a classe TestaConta
- Crie um objeto da classe Conta
- Inicialize todos os atributos deste objeto.
- Imprima os valores dos atributos da classe Cliente de forma a obter o seguinte resultado:

```
* -----  
* AGENCIA: 1   BANCO : 234  
* CONTA CORRENTE : 01945  
* TITULAR: FULANO  
* SALDO : R$10000.0  
* -----
```

Anotações

17

Programação Orientada a Objetos

Exercícios:

Explique os seguintes conceitos :

Objeto:

Classe:

Atributo:

Método:

Anotações

18

Programação Orientada a Objetos

Lista 9

Leia o que está sendo solicitado nos códigos e os implemente.

```
/*  
* 1) Implemente os métodos que não foram implementados na classe Conta de  
acordo com a especificação nos métodos.  
*/
```

```
public class Conta {
```

```
    saldo(double)  
    nrAgencia(String)  
    titular(String)  
    nrConta(String)  
    codBanco(int)
```

```
    // crie o método construtor
```

```
/**  
* @param valor: valor a ser sacado da conta  
* 1. Verificar se o valor do saque e positivo.  
* 2. Verificar se ha saldo suficiente para efetuar o saque  
* 2.1. Se o saldo for suficiente, efetuar o saque  
* 2.2. Se o saldo for insuficiente imprimir na tela que o saldo e Insuficiente  
*/
```

```
public void saque(double valor) {
```

```
}
```

```
/**
```

Anotações

19

Programação Orientada a Objetos

```
* @param valor Valor a ser depositado da conta
* Verificar se o valor do deposito e positivo.
*/
void deposito(double valor) {

}
/**
 * Metodo para impressao de todos os dados da classe
 */
public void imprimeDados() {
    System.out.println("\n-----");
    System.out.println("AGENCIA:\t"+nrAgencia+"\t BANCO:\t"+codBanco);
    System.out.println("Conta: \t"+nrConta);
    System.out.println("TITULAR: \t"+titular);
    System.out.println("SALDO: \t"+saldo);
    System.out.println("-----\n");
}

/**
 * @return saldo da conta
 */
double getSaldo() {
    return saldo;
}
}
```

Anotações

20

Programação Orientada a Objetos

```
public class TestaConta {  
  
    public static void main(String[] args) {  
        // Criacao da conta  
  
        // Inicializacao da conta  
  
        // Impressao dos dados da conta  
  
        // Saque da conta  
  
        // Impressao dos dados da conta  
  
        // Deposito em conta  
  
        // Impressao dos dados da conta  
  
        // Impressao do saldo da conta, utilizando o metodo getSaldo();  
  
    }  
}
```

Anotações

21

Programação Orientada a Objetos

Lista 10

1) Considere o aplicativo chamado **Administracao** e a classe **Populacao**, apresentados a seguir e codifique as questões de 1 a 4:

public class Populacao{

private int pop[][];

public int estados, municipios;

- 1) Codificar neste quadro o construtor, da classe, que recebe como parâmetros o número de estados e o número de municípios, e cria a matriz de populações.

```
public void atualizarPopulacao(int i, int j, int populacao){  
    if (i>=0 && i<4 && j>=0 && j<5 && populacao > 0)  
        pop[i][j] = populacao;  
}
```

- 2) Codificar neste quadro o método que determina a população média de um dado estado.

}

Anotações

22

Programação Orientada a Objetos

```
public class Administracao{  
    public static void main (String p[ ]){
```

3) Declarar variáveis

```
        for (i=0; i<4; i++){  
            for (j=0; j<5; j++){  
                n = Integer.parseInt(JOptionPane.showInputDialog  
                ("Informe a população da cidade " +  
                String.valueOf(j+1) + "\ndo estado " +  
                String.valueOf(i+1));  
                pop.atualizarPopulacao(i, j, n);  
            }  
        }  
    }
```

4) Codificar, neste espaço, a parte do aplicativo que recebe via janela de diálogo o número de um estado e exibe, também via janela de diálogo, a população média deste estado. Utilizar para isto uma matriz de população de 4 estados com 5 municípios cada, gerada através da classe Populacao.

```
    }  
}
```

Anotações

23

Programação Orientada a Objetos

2) Implemente uma classe chamada Carro com as seguintes propriedades:

a. Um veículo tem um certo consumo de combustível (medidos em km/litro) e uma certa quantidade de combustível no tanque.

b. O consumo é especificado no construtor e o nível de combustível inicial é 0.

c. Forneça um método andar() que simule o ato de dirigir o veículo por uma certa distância, reduzindo o nível de combustível no tanque de gasolina.

d. Forneça um método getCombustivel(), que retorna o nível atual de combustível.

e. Forneça um método setCombustivel(), para abastecer o tanque.

f. Escreva um pequeno programa que teste sua classe. Exemplo de uso:
Carro uno(16); // 16 quilômetros por litro de combustível.

uno.setCombustivel(20); // abastece com 20 litros de combustível.

uno.andar(150); // anda 150 quilômetros.

uno.getCombustivel() // Exibe o combustível que resta no tanque.

Anotações

24

Programação Orientada a Objetos

3) Escreva uma classe chamada Aluno que contenha os atributos privados denominados nome, matricula, nota1, nota2, nota3, peso1, peso2 e peso3. Além disso,

a. Crie um construtor-padrão para a classe.

b. Crie um construtor que inicialize todos os membros de dados com os valores recebidos como argumento.

c. Crie os métodos de acesso (*getters* e *setters*) para todos os atributos. Os métodos *setters* devem validar os dados de entrada conforme as regras definidas abaixo:

- o nome deve conter pelo menos dois caracteres e não deve possuir números;

- a matrícula de ser constituída de 9 dígitos apenas;

d. Escreva um método público para calcular a média ponderada das três notas;

Anotações

25

Programação Orientada a Objetos

4) Escreva um programa que leia o nome e salário atual de um funcionário. O programa deve calcular seu novo salário (segundo a tabela abaixo) e mostrar o nome, o salário atual e o salário reajustado do funcionário:

Tabela de Reajuste **Acréscimo**

De	Até	
--	150,00	25%
150,00	300,00	20%
300,00	600,00	15%
600,00	--	10%

- Leia um valor **N** inteiro pelo teclado e realize todo o processo acima descrito para os **N** funcionários;
- mostrar ao final do programa a soma dos salários atuais, a soma dos salários reajustados e a diferença entre eles.

Anotações

26

Programação Orientada a Objetos

5) Crie uma classe chamada **Pessoa**. Uma pessoa possui um nome, idade, peso e altura.

a) implemente o método calculaIMC e apresenta uma mensagem de acordo com a tabela abaixo.

Resultado	Mensagem
Abaixo de 17	Muito abaixo do peso
Entre 17 e 18,49	Abaixo do peso
Entre 18,5 e 24,99	Peso normal
Entre 25 e 29,99	Acima do peso
Entre 30 e 34,99	Obesidade I
Entre 35 e 39,99	Obesidade II (severa)
Acima de 40	Obesidade III (mórbida)

O IMC é uma sigla utilizada para Índice de Massa Corporal, que é uma medida utilizada para medir a obesidade. Fórmula: **imc = peso / altura² ou imc = Peso / (altura*altura)**

b) Crie uma classe TestaPessoa

- Crie um objeto da classe Pessoa
- Inicialize todos os atributos
- Imprima os valores dos atributos da classe Pessoa, o valor do imc e a mensagem de acordo com a tabela acima.

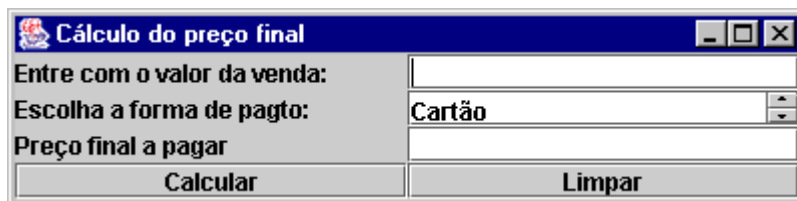
Anotações

27

Programação Orientada a Objetos


Lista 11

1) Crie uma aplicação que simule o cálculo do valor final de uma venda, dependendo da forma de pagamento escolhida pelo usuário. O usuário entra com um valor, escolhe a forma de pagamento e o cálculo do preço final é realizado conforme os seguintes critérios: para pagamento em dinheiro, desconto de 5%, para pagamento em cheque, acréscimo de 5%, para pagamento com cartão, acréscimo de 10%. A figura abaixo apresenta a janela de execução deste exercício.



A janela intitulada "Cálculo do preço final" possui um layout com labels e campos de entrada. O label "Entre com o valor da venda:" está à esquerda de um campo de texto vazio. Abaixo dele, o label "Escolha a forma de pagto:" está à esquerda de um menu suspenso que mostra "Cartão". Abaixo disso, o label "Preço final a pagar" está à esquerda de outro campo de texto vazio. Na base da janela, há dois botões: "Calcular" e "Limpar".

2) Crie uma aplicação que simule o cadastramento de pessoas. O usuário digita o nome e endereço de uma pessoa, escolhe o sexo e o estado civil por meio de componentes do tipo Combo. Ao pressionar o botão mostrar, todos os dados cadastrados são copiados para um componente TextArea, conforme apresenta a figura abaixo.

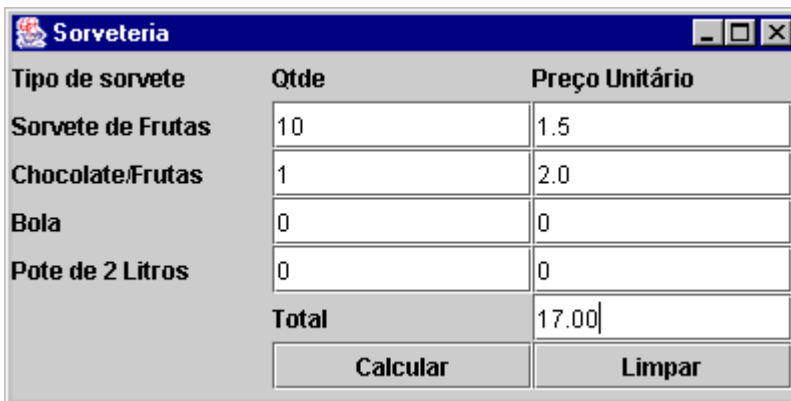


A janela intitulada "Cadastramento de Pessoas" contém campos para "Nome:" (com o texto "Jadir"), "Endereço:" (com o texto "Rua a"), "Sexo" (menu suspenso com "Masculino" selecionado) e "Estado Civil" (menu suspenso com "Casado" selecionado). Abaixo desses campos estão os botões "Mostrar" e "Limpar". Na parte inferior da janela, há uma área de texto (TextArea) que exibe os dados cadastrados: "Nome: Jadir", "Endereço: Rua a", "Sexo: Masculino" e "E.C.: Casado".

Anotações

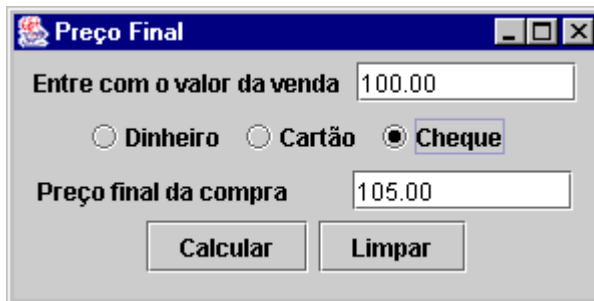
Programação Orientada a Objetos

3) Crie uma aplicação que simule vendas de sorvete em um sorveteria, de acordo com o apresentado na figura abaixo.



Tipo de sorvete	Qtde	Preço Unitário
Sorvete de Frutas	10	1.5
Chocolate/Frutas	1	2.0
Bola	0	0
Pote de 2 Litros	0	0
Total		17.00
Calcular		Limpar

4) Crie uma aplicação que simule o cálculo do valor final de uma venda, dependendo da forma de pagamento escolhida pelo usuário. O usuário entra com um valor, escolhe a forma de pagamento e o cálculo do preço final é realizado conforme os seguintes critérios: para pagamento em dinheiro, desconto de 5%, para pagamento em cheque, acréscimo de 5%, para pagamento com cartão, acréscimo de 10%. A figura abaixo apresenta a janela de execução deste exercício.



Preço Final	
Entre com o valor da venda	100.00
<input type="radio"/> Dinheiro <input type="radio"/> Cartão <input checked="" type="radio"/> Cheque	
Preço final da compra	105.00
Calcular	Limpar

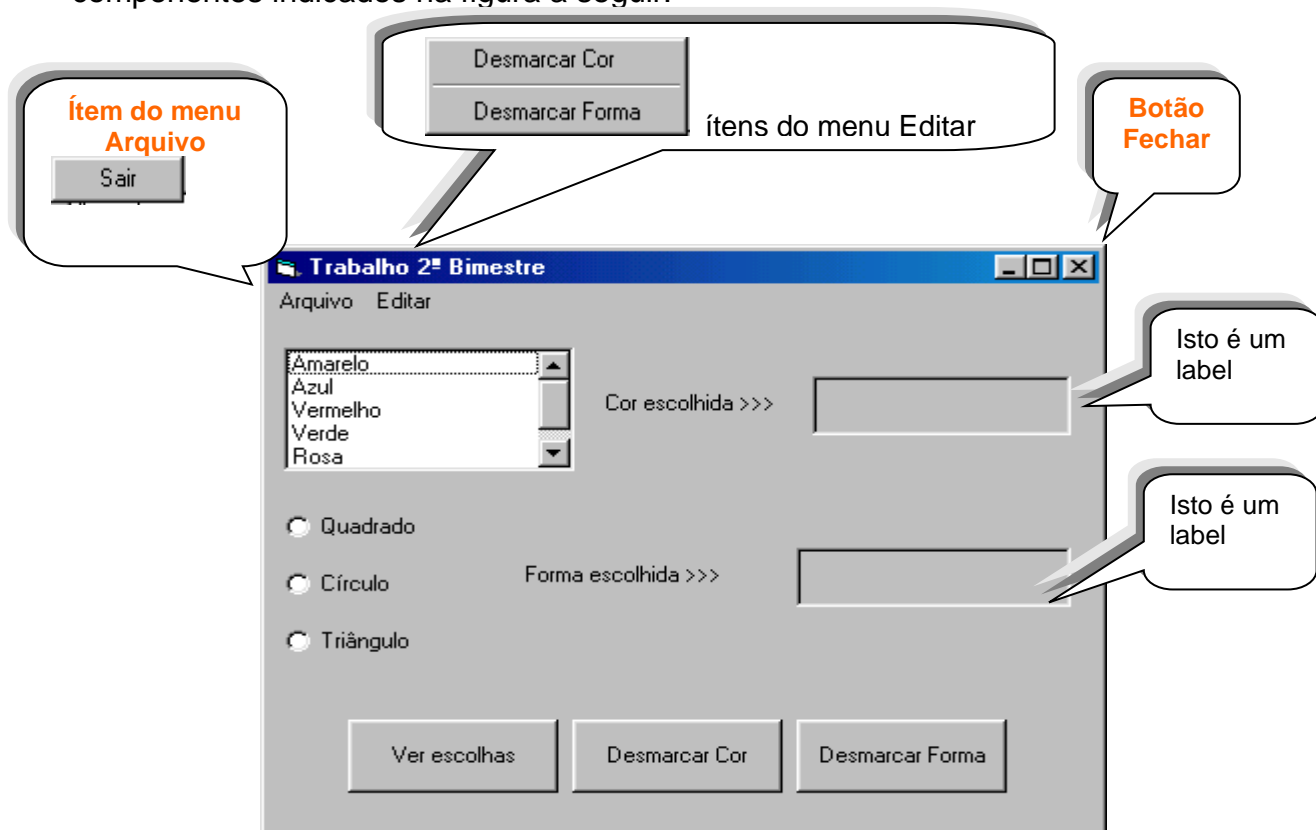
Anotações

29

Programação Orientada a Objetos

Lista 12

1) Codificar uma classe MinhaJanela (extensão de JFrame), que contenha os componentes indicados na figura a seguir:



Observações a respeito da janela:

- ✎ A lista de cores deve ser carregada dinamicamente com cores que serão fornecidas via linha de comando do aplicativo que utilizar esta janela.

Anotações

30

Programação Orientada a Objetos

Lista 13

Implemente um aplicativo usando as Classes Swing e JDBC, conforme a tela abaixo:

IMPORTANTE:

Nome da base de dados: **Biblioteca**

Nome da tabela: **Leitor**

Nomes dos campos:

codLeitor – numérico de 4 bytes

nomeLeitor – texto de 40 bytes

tipoLeitor – texto de 20 bytes

Manutencao do Leitor

Código do Leitor

Nome do Leitor

Tipo de Leitor

Professor
Aluno

Novo Gravar Consultar Alterar Excluir

Banco Conectado com sucesso

Anotações

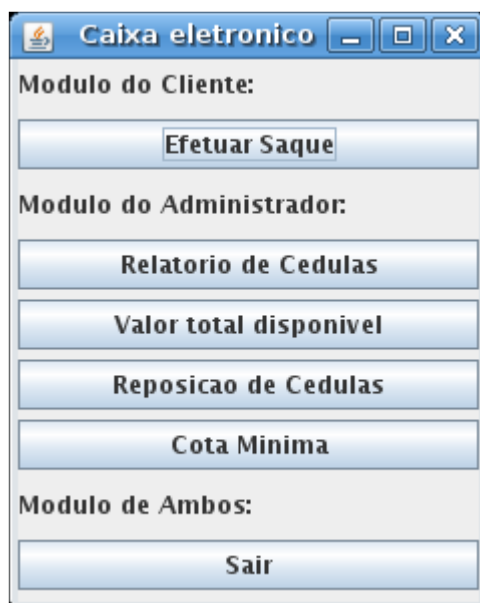
31

Programação Orientada a Objetos

Exercício - Caixa eletrônico

Faça um programa para controlar um caixa eletrônico. Existem 6 tipos de notas: de 2, de 5, de 10, de 20, de 50, de 100. O programa deve inicialmente ler uma quantidade de notas de cada tipo, simulando o abastecimento inicial do caixa eletrônico. Depois disto, o caixa entra em operação contínua atendendo um cliente após o outro. Para sacar, o cliente fornece o valor do saque a ser efetuado e como resultado da operação, o programa deverá então escrever na tela a quantidade de notas de cada tipo que será dada ao cliente a fim de atender ao seu saque. Sempre que um saque for efetuado por um cliente, a quantidade inicial de dinheiro que foi colocada no caixa é decrementada. O programa deve pagar sempre com as maiores notas possíveis. Sempre que não for possível pagar somente com notas de 100, então o programa tentará complementar com notas de 50, depois com notas de 20, 10, 5 e 2. Antes de efetuar um saque para um cliente, ou seja, escrever na tela as notas que ele irá receber, o programa deve ter certeza que é possível pagá-lo, senão emitirá uma mensagem do tipo “Não Temos Notas Para Este Saque”. Caso o caixa fique abaixo de um certo mínimo, o algoritmo deverá parar de atender aos clientes e emitir uma mensagem do tipo “Caixa Vazio: Chame o Operador”.

A interface com usuário (figura 1) do caixa eletrônico já é fornecido para você, juntamente com um contrato (Programa 2) para utilização da interface, que segue abaixo.



Anotações

32

Programação Orientada a Objetos

Figura 1 _ interface de utilização do caixa eletrônico

```
/**
 * Interface (contrato) para utilizacao da interface grafica.
 * Nesse contrato e definido as operacoes de entrada e saida de dinheiro
 do caixa eletronico
 */

public interface ICaixaEletronico{
/**
 * Pega o valor total disponivel no caixa eletronico
 * @retorna uma string formatada com o valor total disponivel
 */
public String pegaValorTotalDisponivel();
/**
 * Efetua o saque
 * @param valor a ser sacado
 * @retorna uma string formatada informando o resultado da operacao
 */
public String sacar(Integer valor);
/**
 * Pega um relatorio informando as celulas e a quantidade de celula
 disponivel
 * @retorna uma string formatada com as celula e suas quantidades
 */
public String pegaRelatorioCedulas();
/**
 * Efetua a reposicao de cédulas
 * @param cedula de reposicao
 * @param quantidade de cédulas para reposicao
 * @retorna uma string formatada informando o resultado da operacao
 */
public String reposicaoCedulas(Integer cedula, Integer quantidade);
/**
 * Efetua a leitura da cota minima de atendimento
 * @param minimo
 * @retorna uma string formatada informando o resultado da operacao
 */
public String armazenaCotaMinima(Integer minimo);
}
```

Programação Orientada a Objetos

Programa 2 - contrato de utilização da interface gráfica

Seu programa deve criar um classe chamado CaixaEletronico e implementar o contrato definido em IcaixaEletronico, como segue abaixo:

```
public class CaixaEletronico implements ICaixaEletronico{
    public String pegaRelatorioCedulas() {
        String resposta = "";
        //logica de fazer o relatorio de cedulas
        return resposta;
    }
    public String pegaValorTotalDisponivel() {
        String resposta = "";
        //logica de pega o valor total disponivel no caixa eletronio
        return resposta;
    }
    public String reposicaoCedulas(Integer cedula, Integer quantidade) {
        String resposta = "";
        //logica de fazer a reposicao de cedulas e criar uma mensagem
        //(resposta)ao usuario
        return resposta;
    }
    public String sacar(Integer valor) {
        String resposta = "";
        //logica de sacar do caixa eletronico e criar um mensagem(resposta) ao //
        usuario
        return resposta;
    }
    public String armazenaCotaMinima(Integer minimo) {
        String resposta = "";
        //logica de armazenar a cota minima para saque e criar um
        //mensagem(resposta)ao usuario
        return resposta;
    }
    public static void main(String arg[]){
        GUI janela = new GUI(CaixaEletronico.class);
        janela.show();
    }
}
```

Anotações

34

Programação Orientada a Objetos

Programa 3 _ Classe *CaixaEletronico* implementando o contrato com *ICaixaEletronico*. Observe que o método *main* de *CaixaEletronico* já está se comunicando com a interface gráfica.

Essa classe deve trabalhar com uma matriz 6 x 2, responsável por guardar a quantidade de cédulas disponível de cada valor. Veja a tabela a seguir:

Coluna 0 (valor das células)	Coluna 1 (quantidade de cédulas)
100	100
50	200
20	300
10	350
5	450
2	500

- O botão **Efetuar saque** deve fazer uma simulação de saque no caixa eletrônico. Quando o usuário escolher esta opção, o programa deverá solicitar o valor do saque e em seguida efetuar o saque , mostrando na tela quantas cédulas de cada valor foram emitidas.
 - O programa deve fazer o cálculo de quais cédulas serão emitidas visando emitir o menor número de notas possível, dando prioridade para as cédulas de maior valor. Para simular o saque, o programa deve fazer a devida atualização na matriz de quantidades de cédulas disponíveis.
 - Se as notas de algum valor acabarem, o programa deve tentar efetuar o saque através das demais notas existentes, caso seja possível, sempre visando emitir o menor número de cédulas.
 - Se não for possível a realização do saque solicitado com a quantidade de notas existentes, o programa deverá emitir a mensagem **“Saque não realizado por falta de cédulas”**.
 - O programa não deverá permitir que mais de 30 cédulas sejam emitidas, impossibilitando os saques nesses casos.

- O botão **Relatório Cédulas** o programa deverá mostrar a matriz de

Anotações

35

Programação Orientada a Objetos

quantidades de cédulas, informando quantas notas estão disponíveis para cada valor no compartimento.

- O botão **Valor total disponível** deverá apresentar o valor total em reais disponível no caixa.
- O botão **Reposição de Cédulas** deve possibilitar que o usuário faça a reposição das cédulas.
- O botão **Conta Mínima** deve possibilitar armazenar o valor da conta mínima. Caso o caixa fique abaixo da cota mínima, o algoritmo deverá parar de atender aos clientes e emitir uma mensagem do tipo “**Caixa Vazio: Chame o Operador**”.

IMPORTANTE:

Ao clicar no botão **sair** deve ser apresentado um extrato com todos os saques e atualização de saldo, cada grupo é responsável pelo layout do extrato.

Regras de entrega do Projeto:

A data de entrega será definida em sala de aula. Não será aceito trabalho entregue fora do prazo.

Deverá ser entregue uma cópia impressa por grupo, sem a cópia o grupo não poderá apresentar o trabalho.

Essa atividade pode ser feita em no máximo 5 alunos.

Projetos iguais serão considerados como cola, portanto não será aceito;

Anotações

36

Programação Orientada a Objetos

Regras para correção:

Programas incompletos ou que não estejam compilando não serão aceitos;

Nota final será composta por:

50% da nota para uma pergunta individual respondida corretamente sobre o Projeto. (avaliação oral)

10% da nota para as 4 perguntas respondida corretamente pelo grupo

20% da nota para documentação do projeto. Comentários no código fonte e organização do código.

20% da nota será destinado a funcionalidade do projeto

Regras para implementação dos Programas:

Não será aceito uso de bibliotecas externas.

Vocês não precisaram construir interface com usuário. A interface esta sendo fornecida.

Vocês precisaram implementar a interface ICaixaEletronico fornecida para testar seu Projeto com uma interface gráfica disponível;

Vocês não poderão mudar os métodos da interface ICaixaEletronico;

Na ICaixaEletronico está documentado exatamente o que cada método deverá fazer;

Vocês poderão conversar sobre o problema mas não poderão trocar códigos, isso poderá acarretar em nota zero para o Projeto.

Anotações

37

Classe Agenda

```
/* Crie uma classe chamada Pessoa. Uma pessoa possui um nome
e uma idade.
- crie 2 construtores: 1 que recebe o nome e a idade como
parâmetros de entrada e um que não recebe parâmetros e inicializa
os atributos com um valor padrão ("indefinido" para Strings e 0
para inteiros).
- crie os métodos de acesso para os atributos (GET e SET). */

/* Crie uma classe Amigo, que herda Pessoa, e possui uma data
de aniversário.
- crie um construtor que não recebe parâmetros de entrada, e
inicializa o atributo com um valor padrão ("indefinido", por
exemplo).
- crie os métodos de acesso para o atributo data de nascimento.
*/

/* Crie uma classe Conhecido, que herda Pessoa, e possui um email.
- crie um construtor que não recebe parâmetros de entrada, e
inicializa o e-mail com um valor padrão ("indefinido", por
exemplo).
- crie os métodos de acesso para este atributo. */

/* Crie agora, uma classe Agenda, que possui pessoas (em um array)
e dois atributos que controlam: a quantidade de amigos e a
quantidade de conhecidos.
- crie um construtor que recebe por parâmetro a quantidade de
pessoas que a agenda terá, e inicializa o array de Pessoa. Neste
construtor, inicialize todas as posições do array criando
ALEATORIAMENTE um Conhecido ou um Amigo (utilize o comando:
1 + (int) (Math.random() * 2)
para sortear valores entre 1 e 2. Se o valor encontrado for 1,
crie um Amigo. Se o valor encontrado for 2, crie um Conhecido).
- crie os métodos GET para todos os atributos da classe Agenda.
- crie um método chamado addInformacoes, que não recebe parâmetros
de entrada. Para cada Pessoa na agenda, peça para o usuário
```

38

Anotações

Programação Orientada a Objetos

digitar (via teclado) as informações cabíveis para cada tipo de Pessoa, e acesse os métodos SET para atribuir as informações.

- crie um método chamado imprimeAniversários, que imprime os aniversários de todos os amigos que estão armazenados na agenda.
- crie um método chamado imprimeEmail, que imprime os e-mails de todos os conhecidos que estão armazenados na agenda. */

/* Crie uma classe de teste para a Agenda.

- peça para o usuário informar (via teclado) quantas pessoas ele deseja colocar na agenda, e crie uma Agenda com esta informação.
 - imprima na tela a quantidade de amigos e de conhecidos na agenda.
 - adicione informações à agenda.
 - imprima todos os aniversários dos amigos presentes na agenda.
 - imprima todos os e-mails dos conhecidos armazenados na agenda.
- */