

Iniciado em terça, 14 Jun 2022, 11:00

Estado Finalizada

Concluída em sábado, 18 Jun 2022, 11:21

Tempo 4 dias

empregado

Avallar 10,00 de um máximo de 10,00(100%)

Questão 1
Correto
Atingiu 2,00 de 2,00
Marcar questão

### Banco

Construa um programa para simulação de um banco que manipula contas, responsável pela criação das contas e suas operações. Crie a classe "Conta" com os atributos privados numeroConta (int), nomeCliente (String), saldo (double) e senha (int). Considere que o banco não oferece limite ao cliente e não realiza operações de saque ou transferência caso não tenha valor suficiente no saldo (ou seja, a conta não ficará negativa). A classe Conta deve ter um método construtor considerando todos os atributos e os métodos getters considerando os atributos numeroConta, nomeCliente e saldo. No método get para retornar o saldo, considere primeiro uma verificação da senha conforme o valor recebido por parâmetro. Além destes métodos, a classe Conta também deve conter três métodos públicos com retornos do tipo boolean, sendo um método para sacar um valor da conta, um método para depositar um valor na conta e um método para fazer a transferência entre duas contas, com as seguintes funcionalidades:

- O método para fazer saque da conta deve receber dois argumentos, o valor do saque (double) e a senha da conta (int). O método deve retirar o valor da conta considerando o atributo saldo, caso o valor for maior do que zero, menor que o saldo atual e a senha for válida (retorno true); caso contrário o retorno é falso.
- O método para fazer depósito na conta deve receber dois argumentos, o valor do depósito (double) e a senha da conta (int). O método deve adicionar o valor na conta considerando o atributo saldo, caso o valor for maior do que zero e a senha for válida (retorno true); caso contrário o retorno é falso.
- O método para fazer transferência entre contas deve receber três argumentos, o valor da transferência (double), a senha da conta (int) e a conta2 que receberá a transferência (objeto do tipo Conta). O método deve retirar o valor da conta considerando o atributo saldo e adicionar o valor na conta2, caso o valor for maior do que zero, menor que o saldo atual e a senha for válida (retorno true); caso contrário o retorno é falso.

Crie a classe Banco (classe principal) e instancia dois objetos do tipo conta (conta1 e conta2) usando o construtor para inicializar os valores da conta, considerando os dados de entrada pelo teclado. Em seguida, o usuário da conta1 poderá escolher entre cinco opções de operação (int): visualizar saldo, sacar valor, depositar valor, transferência entre contas e sair do sistema. Considere a possibilidade de o usuário fazer repetidas operações enquanto não sair do sistema (usar estrutura de repetição). Considere as seguintes funcionalidades para cada opção de operação:

1. visualizar saldo: solicite a senha da conta e acesse o método da classe Conta para retornar o saldo caso a senha esteja correta; caso contrário apresentar a mensagem "senha incorreta";
2. sacar valor: solicite o valor do saque e senha da conta, e acesse o método da classe Conta para sacar; apresente a mensagem "saque realizado" ou "saque não realizado" de acordo com o retorno do método;
3. depositar valor: solicite o valor do depósito e senha da conta, e acesse o método da classe Conta para depositar; apresente a mensagem "depósito realizado" ou "depósito não realizado" de acordo com o retorno do método;
4. transferência entre contas: solicite o nome do cliente que usuário deseja fazer a transferência de dinheiro; verifique se o nome do cliente é o mesmo do objeto da conta2 usando o método de retorno do nome do cliente; se os nomes forem diferentes, apresentar uma mensagem "nenhum usuário encontrado"; caso contrário solicite o valor da transferência e senha da conta, e acesse o método da classe Conta para transferir; apresente a mensagem "transferência realizada" ou "transferência não realizada" de acordo com o retorno do método;
5. sair do sistema: finalize o programa.

#### Entrada:

As duas primeiras linhas são informações de duas contas, uma em cada linha. Cada conta tem quatro valores separados por espaço: número da conta (int), senha da conta (int), nome do cliente (String - apenas o primeiro nome), saldo inicial de abertura da conta (double). A próxima linha é um valor inteiro (de 1 a 5) que corresponde a opção do menu que o cliente deseja realizar (não é necessário validar a entrada). Enquanto o valor 5 não for informado, serão solicitadas as entradas de acordo com cada opção. A opção 1 tem apenas uma entrada, que é a senha da conta1 (int). A opção 2 tem duas entradas separadas por espaço, que são o valor do saque (double) e a senha da conta1 (int). A opção 3 tem duas entradas separadas por espaço, que são o valor do depósito (double) e a senha da conta1 (int). A opção 4 tem duas linhas de entrada. Na primeira linha, o nome do cliente para a transferência (String). Se o nome estiver correto, serão solicitados dois valores separados por espaço na linha seguinte, que são o valor da transferência (double) e a senha da conta1 (int). A opção 5 não tem nenhuma entrada.

#### Saída:

Para a opção 1, a saída é um valor double com duas casas decimais ou a mensagem "senha incorreta" dependendo do retorno do método. Para as opções 2, 3 e 4, a saída é uma mensagem conforme apresentada no enunciado do problema. Para a opção 5 não tem nenhuma saída.

#### Objetivos de estudo:

- Comunicação entre dois objetos;
- Considere testes fora do coderunner para analisar outros exemplos e a criação de vetor para armazenar vários

### Navegação do questionário

1	2	3	4	5
✓	✓	✓	✓	✓

[Mostrar uma página por vez](#)[Terminar revisão](#)

objetos do tipo conta e a definição utilização de um atributo limite para a classe conta).

For example:

Input	Result
000001 123 Maria 500.00	500.00
000002 321 Pedro 200.00	saque realizado
1	400.00
123	
2	
100 123	
1	
123	
5	

Answer: (penalty regime: 0, 0, 0, 1, 2, ... %)

```
1 import java.util.*;
2
3 class Conta {
4     private int numeroConta;
5     private String nomeCliente;
6     private double saldo;
7     private int senha;
8
9     public Conta(int numeroConta, int senha, String nomeCliente, double saldo) {
10         this.numeroConta = numeroConta;
11         this.senha = senha;
12         this.nomeCliente = nomeCliente;
13         this.saldo = saldo;
14     }
15 }
```

	Input	Expected	Got	
✓	000001 123 Maria 500.00 000002 321 Pedro 200.00 1 123 2 100 123 1 123 5	500.00 saque realizado 400.00	500.00 saque realizado 400.00	✓
✓	000005 432 Ana 200.00 000006 589 Paula 800.00 1 589 5	senha incorreta	senha incorreta	✓
✓	000010 123 Carlos 1000.00 000011 456 Victor 5.00 2 2000 123 2 500 321 5	saque não realizado saque não realizado	saque não realizado saque não realizado	✓
✓	000009 123 Beatriz 500.00 000010 456 Paola 500.00 3 0 123 3 100 123 5	depósito não realizado depósito realizado	depósito não realizado depósito realizado	✓
✓	000001 123 João 500.00 000002 789 José 700.00 4 Pedro 4 José 200 123 1 123 5	nenhum usuário encontrado transferência realizada 300.00	nenhum usuário encontrado transferência realizada 300.00	✓

Passou em todos os testes! ✓

Correto

Notas para este envio: 2,00/2,00.

## Questão 2

Correto

Atingiu 2,00 de 2,00

Marcar questão

## Círculo

Escreva, compile e analise as classes Círculo e TestaCírculo fora do CodeRunner. Teste a criação de objetos da classe Círculo executando a classe TestaCírculo.

```
class Círculo {
    private int x,y,raio;
    public static final double PI = 3.14159;
    public Círculo(int x, int y, int raio) {
        this.x = x;
        this.y = y;
        this.raio = raio;
    }
    public int getX() {
        return x;
    }
    ...
}
```

```
public int getX() {
    return x;
}

public int getY() {
    return y;
}

public void setRaio(int raio) {
    this.raio = raio;
}

public double circunferencia(){
    return 2 * PI * raio;
}

}

public class TestaCirculo {

    public static void main(String[] args) {

        Circulo circulo1,circulo2,circulo3;
        circulo1 = new Circulo(3, 3, 1);
        circulo2 = new Circulo(2, 1, 4);
        circulo3 = circulo1; //mesmo objeto

        System.out.println(circulo1.getX() + " " + circulo1.getY());
        System.out.println(circulo2.getX() + " " + circulo2.getY());
        System.out.println(circulo3.getX() + " " + circulo3.getY());

        int circulo = (int) circulo1.circunferencia();
        System.out.print(circulo1.getRaio()); //raio do circulo1
        System.out.println(" " + circulo); //circunferência do circulo1
    }
}
```

Modifique a classe TestaCirculo de acordo com as seguintes instruções:

1. Crie um vetor de 3 objetos Circulo, considerando o construtor da classe;
  2. Declare outra referência do tipo Circulo[];
  3. Copie a referência do primeiro vetor para o segundo;
  4. Crie um terceiro vetor do tipo Circulo de tamanho 3;
  5. Copie os objetos do primeiro vetor para o terceiro;
  6. Altere os valores de raio para os objetos do primeiro vetor;
  7. Imprima os valores x, y e raios de cada objeto para os três vetores.

**Entrada:**

Três linhas, cada linha corresponde a um objeto do tipo Círculo, que possui três valores (Int) separados por espaço, que são o x, y e o ralo (Instrução 1). Em seguida, três valores inteiros, um valor em cada linha, que representam valores de raios para cada objeto do tipo Círculo (Instrução 6).

**Saída:**

Três vetores, cada vetor é apresentado em quatro linhas. A primeira linha um texto com a identificação do vetor, conforme a mensagem do exemplo, e as próximas linhas são os valores (int) dos objetos, um objeto em cada linha. Para cada objeto, são apresentados os valores x, y e ralo, separados por espaço.

## Objetivos de estudo:

- Criação e manipulação de vetores de objetos. Analise os resultados;
  - Considere testes fora do coderunner para analisar outros **exemplos**.

For example:

Input	Result
3 3 1	vector1:
4 3 2	3 3 2
6 2 3	4 3 1
1	vector2:
4	3 3 2
	4 3 1
	6 2 4
	vector3:
	3 3 1
	4 3 2
	6 2 3

**Answer:** (penalty regime: 0, 0, 0, 1, 2, ... %)

```
1 v import java.util.*;
2
3 v class Circulo {
4     private int x, y, raio;
5     public static final double PI = 3.14159;
6
7 v     public Circulo(int x, int y, int raio) {
8         this.x = x;
9         this.y = y;
10        this.raio = raio;
11    }
12
13 v     public int getx() {
14         return x;
```

Input	Expected	Got
✓ 3 3 1 4 3 2 6 2 3 2 1 4	vetor1: 3 3 2 3 3 1 4 3 1 6 2 4 vetor2: 3 3 2 4 3 1 6 2 4 vetor3: 3 3 1 3 3 2 6 2 3	✓ vetor1: 3 3 2 3 3 1 4 3 1 6 2 4 vetor2: 3 3 2 4 3 1 6 2 4 vetor3: 3 3 1 3 3 2 6 2 3
3 3 2 3 3 1 4 3 1 6 2 4 vetor2: 3 3 2 4 3 1 6 2 4 vetor3: 3 3 1 3 3 2 6 2 3		
2 1 4		
1 4		

Passou em todos os testes! ✓

Correto

Notas para este envio: 2,00/2,00.

### Questão 3

Correto

Atingiu 2,00 de 2,00

Markar questão

### Conversão de Temperaturas

Crie uma classe "Temperatura" que conterá um atributo privado de temperatura (double) e quatro métodos: um método construtor para a inicialização do valor do atributo, um método `toString` para exibir o valor do atributo, um método para a conversão entre graus Celsius e Fahrenheit ( $\text{temperatura} * 9.0 / 5.0 + 32$ ) e um método para a conversão entre graus Fahrenheit e Celsius ( $(\text{temperatura} - 32) * 5.0 / 9.0$ ). Os métodos de conversão armazenam no atributo da classe a temperatura atual convertida e não retorna nada.

Crie a classe "Converte" (classe principal), que instancia um objeto da classe "Temperatura" com o valor inicial de temperatura em Celsius informado pelo usuário. Em seguida, acesse o método para converter a temperatura do objeto Fahrenheit e a exiba usando o método `toString`. Depois converta para Celsius usando o método correspondente e a exiba novamente usando o método `toString`.

#### Entrada:

Um valor de entrada do tipo double, que representa uma temperatura em Celsius.

#### Saída:

Duas linhas de saída. Na primeira linha, um valor double, que corresponde a temperatura em Fahrenheit e na segunda linha um valor double em Celsius. Ambos os resultados devem ser apresentados com acompanhamento de texto conforme o exemplo.

#### Objetivos de estudo:

- Inicializar um atributo utilizando construtor;
- Alterar o valor de um atributo por meio dos métodos da classe;
- Usar o método `toString` para apresentar os valores de um objeto em forma de texto;
- Analisar o comportamento da chamada dos métodos (considere o teste de outros exemplos fora do coderunner).

For example:

Input	Result
30	temperatura: 86.0 graus fahrenheit temperatura: 30.0 graus celsius

Answer: (penalty regime: 0, 0, 0, 1, 2, ... %)

```
1 import java.util.*;
2
3 class Temperatura {
4     private double temperatura;
5
6     public Temperatura(double temp) {
7         this.temperatura = temp;
8     }
9
10    public void celsiusToFahrenheit() {
11        this.temperatura = temperatura * 1.8 + 32;
12    }
13
14    public void fahrenheitToCelsius() {
15        this.temperatura = (temperatura - 32) * 5.0 / 9.0;
16    }
17
18    @Override
19    public String toString() {
20        return "temperatura: " + this.temperatura;
21    }
22}
```

Input	Expected	Got
✓ 30	temperatura: 86.0 graus fahrenheit temperatura: 30.0 graus celsius	temperatura: 86.0 graus fahrenheit temperatura: 30.0 graus celsius ✓

Passou em todos os testes! ✓

Correto

Notas para este envio: 2,00/2,00.

### Questão 4

Correto

Atingiu 2,00 de 2,00

### Formas

Crie uma classe "Retangulo", sem definir construtor, com dois atributos privados do tipo int, "comprimento" e

2,00

Marcar questão

"largura", configurados **Inicialmente com o valor padrão 1**. A classe deve conter dois métodos públicos e com retorno do tipo int, um para calcular e retornar o perímetro de um retângulo ( $2 * (\text{comprimento} + \text{largura})$ ) e o outro para calcular e retornar a área de um retângulo ( $\text{comprimento} * \text{largura}$ ). A classe Retangulo deve conter também métodos *setters* e *getters* para os atributos comprimento e largura. Os métodos *setters*, antes de atribuir os novos valores aos atributos, também devem verificar se os valores de comprimento e largura são, cada um, números inteiros maiores que 0 e menores que 20, caso contrário não faz a atribuição.

A seguir, crie uma classe "Formas" (classe Principal) que irá instanciar dois objetos do tipo Retangulo. Considere a chamada dos métodos *setters* para alterar os valores dos atributos comprimento e largura para cada objeto de acordo com as informações de entrada. Em seguida, use os métodos *getters* para apresentar os valores dos atributos de cada um desses objetos e também os métodos para calcular e apresentar os seus perímetros e áreas.

**Entrada:**

Duas linhas de entrada. Cada linha contém dois valores inteiros (separados por espaço) para um objeto do tipo Retangulo, que são o comprimento e a largura (nesta ordem).

**Saída:**

Duas linhas de saída. Cada linha contém quatro valores inteiros (separados por espaço) representando informações de um objeto do tipo Retangulo, que são o comprimento, a largura, o perímetro e área (nesta ordem).

**Objetivos de estudo:**

- Inicializar um atributo com um valor padrão;
- Criar vários objetos;
- Trabalhar com atributos privados (declaração);
- Acessar atributos privados na classe principal;
- Analisar o comportamento da chamada dos métodos (considere o teste de outros [exemplos](#) fora do coderunner).

**For example:**

Input	Result
2 3	2 3 10 6
10 21	10 1 22 10

**Answer:** (penalty regime: 0, 0, 0, 1, 2, ... %)

```

1 * import java.util.*;
2
3 * class Retangulo {
4     private int comprimento = 1;
5     private int largura = 1;
6
7     public int perimetro() {
8         int perimetro = 2 * (comprimento + largura);
9         return perimetro;
10    }
11
12    public int area() {
13        int area = comprimento * largura;
14        return area;
15    }

```

	Input	Expected	Got	
✓	2 3 10 21	2 3 10 6 10 1 22 10	2 3 10 6 10 1 22 10	✓

Passou em todos os testes! ✓

**Correto**

Notas para este envio: 2,00/2,00.

**Questão 5**

Correto

Atingiu 2,00 de 2,00

Marcar questão

**Simulação de Impressora**

Crie a Classe "Impressora", sem definir construtor, e implemente os seguintes atributos e métodos públicos:

- Atributos: modelo (String), cor (String), tipoPapel (String), bluetooth (boolean), wifi (boolean), colorida (boolean), ligada (boolean), papel (boolean), digitalizadora (boolean), copiadora (boolean);
- Métodos (void):
  - ligar (se o status do atributo ligada estiver false, mostrar a mensagem "ligando..." e alterar o atributo ligada para true; caso contrário, não fazer nada);
  - desligar (se o status do atributo ligada estiver true, mostrar a mensagem "desligando..." e alterar o atributo ligada para false; caso contrário, não fazer nada);
  - imprimir (se o status do atributo ligada estiver true e se o status do papel estiver true, mostrar a mensagem "imprimindo..."; senão se o status do atributo ligada estiver true e se o status do papel estiver false, mostrar a mensagem "sem papel"; caso contrário mostrar a mensagem "Impressora desligada");
  - digitalizar (se os status dos atributos ligada e digitalizadora estiverem true, mostrar a mensagem "digitalização sendo realizada..."; senão se o status de ligada estiver true e o status de digitalizadora false, mostrar a mensagem "não é possível digitalizar"); caso contrário mostrar a mensagem "Impressora desligada");
  - copiar (se os status dos atributos ligada e copiadora estiverem true, mostrar a mensagem "cópia sendo realizada..."; senão se o status de ligada estiver true e o status de copiadora false, mostrar a mensagem "não é possível copiar"); caso contrário mostrar a mensagem "Impressora desligada");

- possivel copiar ");, caso contrário mostrar a mensagem "Impressora desligada";
- status (mostrar o valor de todos os atributos, não esqueça de tratar os atributos lógicos; mensagens para atributos com valores true ou false: "bluetooth on" ou "bluetooth off" (atributo bluetooth), "wifi on" ou "wifi off" (atributo wifi), "Impressão colorida" ou "impressão preto e branco" (atributo colorida), "impressora on" ou "impressora off" (atributo ligada), "tem papel" ou "não tem papel" (atributo papel), "digitalizadora on" ou "digitalizadora off" (atributo digitalizadora), "copiadora on" ou "copiadora off" (atributo copiadora).

Crie a Classe Principal "Usalmpressora" e instancie um objeto da classe Impressora, "impressora". Considerando o objeto criado, atribua valores iniciais a todos os atributos lógicos definidos na classe (todos devem receber valor true, com exceção dos atributos ligada e copiadora). Em seguida, solicite ao usuário o modelo, a cor e o tipo de papel da impressora. Chame o método status, para imprimir na tela todas as características do objeto. Por fim, para analisar o comportamento dos métodos, acesse os métodos ligar, digitalizar, copiar, desligar e imprimir (nesta ordem).

#### Entrada:

Três entradas do tipo String: o modelo da impressora, a cor da impressora e o tipo de papel que a impressora suporta. Cada entrada é composta por apenas uma palavra e todas as entradas são informadas na mesma linha (Dica: <entrada>.next()).

#### Saída:

As saídas serão de acordo com os dados atuais armazenados nos atributos e as mensagens definidas nos métodos. Cada saída é apresentada em uma linha (dica: System.out.println()), considerando esta ordem:

- O modelo da impressora;
- A cor da impressora;
- O tipo de papel que a impressora suporta;
- Status do bluetooth;
- Status do wifi;
- Status da cor de impressão (colorida ou em preto e branco);
- Status se a impressora está ligada ou não;
- Status se a impressora tem papel ou não;
- Status se a impressora está no modo digitalizadora ou não;
- Status se a impressora está no modo copiadora ou não;
- Mensagem do método ligar;
- Mensagem do método digitalizar;
- Mensagem do método copiar;
- Mensagem do método desligar;
- Mensagem do método imprimir.

#### Objetivos de estudo:

- Criar uma classe e instanciá-la no método principal;
- Trabalhar com atributos e métodos públicos (declaração);
- Diferenciar as inicializações de atributos por leitura do teclado e por atribuição com valores fixos;
- Acessar atributos e métodos públicos na classe principal;
- Analisar o comportamento da chamada dos métodos (considere o teste de outros [exemplos](#) fora do coderunner).

#### For example:

Input	Result
HP preta A4	HP preta A4 bluetooth on wifi on impressão colorida impressora off tem papel digitalizadora on copiadora off ligando... digitalização sendo realizada... não é possível copiar desligando... impressora desligada

**Answer:** (penalty regime: 0, 0, 0, 1, 2, ... %)

```

1 * import java.util.*;
2
3 * class Impressora {
4     public String modelo, cor, tipoPapel;
5     public boolean bluetooth, wifi, colorida, ligada, papel, digitalizadora, copiadora;
6
7     public void ligar() {
8         if (!ligada) {
9             System.out.println("ligando...");
10            ligada = true;
11        }
12    }
13
14     public void desligar() {
15         if (ligada)
16             System.out.println("desligando...");
17     }
18 }
```

Input	Expected	Got
-------	----------	-----

✓	HP preta A4	HP preta A4 bluetooth on wifi on impressão colorida impressora off tem papel digitalizadora on copiadora off ligando... digitalização sendo realizada... não é possível copiar desligando... impressora desligada	HP preta A4 bluetooth on wifi on impressão colorida impressora off tem papel digitalizadora on copiadora off ligando... digitalização sendo realizada... não é possível copiar desligando... impressora desligada	✓
---	-------------	---	---	---

Passou em todos os testes! ✓

Correto

Notas para este envio: 2,00/2,00.

[Terminar revisão](#)



ATIVIDADE ANTERIOR

Atividade 3.1 - Lista de exercícios teóricos - Java

PRÓXIMA ATIVIDADE

Atividade 3.3 - Lista extra - C++



[Seguir para...](#)



[Obter o aplicativo para dispositivos móveis](#)