INSTITUTO FEDERAL Bahia Campus Vitória da Conquista	**AVALIAÇÃO 2**	
	CURSO: Bacharelado em Sistemas de Informação	**MODALIDADE:** Ensino Superior
	MÓDULO/SEMESTRE/SÉRIE: 3º	**PERÍODO LETIVO:** 2024.1
	DISCIPLINA: Linguagem de Programação II	**CLASSE:** 20241.3.119.1N
	DOCENTE: Alexandro dos Santos Silva	
	DISCENTE:	**CONCEITO:**

INSTRUÇÕES

- A avaliação é composta de 4 (quatro) questões com escore total de 5 (cinco) pontos;
- **Em caso de identificação de plágio, todos os discentes envolvidos terão sua avaliação anulada, com consequente atribuição de conceito nulo**;
- Para resolução das questões, será admitido o uso apenas da sintaxe adotada para escrita de programas em Java;
- Será admitida entrega da resolução apenas através do ambiente da disciplina junto à plataforma Google Sala de Aula e em prazo estabelecido pelo docente.
- Para efeito de entrega da resolução, **todos os arquivos de extensão .java gerados devem ser compactados em um único arquivo, cujo nome deve corresponder ao próprio nome do discente**, sob pena, inclusive, de penalização.

---

1. (Peso: 1,5) A caderneta de poupança é um tipo de investimento de renda fixa com rentabilidade padronizada definida pelo governo através do Banco Central do Brasil. Na prática, a quantia é depositada em uma conta poupança e, a cada mês, é aplicado um percentual de rendimento sobre aquele valor depositado. Essa remuneração, contudo, é creditada apenas na data de "aniversário", que nada mais é do que o dia do mês em que o depósito foi feito. A título de exemplificação, caso uma pessoa tenha depositado a quantia no dia 22 (vinte e dois) de março, a rentabilidade e o crédito correspondente na conta daquele pessoa ocorrerá somente no dia 22 (vinte e dois) dos meses posteriores. No quadro abaixo, estão listadas taxas de rendimento da poupança nos últimos 6 (seis) meses, conforme dados disponíveis em **http://www.ipeadata.gov.br/ExibeSerie.aspx?serid=31878&Module=M**:

Mês	Taxa de Rendimento (%)
09/2023	0,61
10/2023	0,61
11/2023	0,58
12/2023	0,57
01/2024	0,59
02/2024	0,5

Em caso de abertura de uma caderneta de poupança no valor de R$ 125,00 no dia 22/09/2023 e de modo a não haver nenhum depósito posterior, o saldo atualizado da caderneta no dia 22/03/2024 seria de exatos R$ 129,38 após crédito, mês a mês, das renumerações de acordo com as taxas de rendimento mostradas no quadro acima **(rendimento acumulado no período seria de exatos de R$ 4,38, o que corresponde a uma taxa de rendimento acumulada de aproximadamente 3,50%)**.

Posto isto, segue-se abaixo classe para fins de encapsulamento de dados de cadernetas de poupança, de nome `CadernetaPoupanca`.

```
01  package lingprog2.aval2.versao1.questao1;
02
03  public class CadernetaPoupanca {
04
05      private String titular;               // nome do titular
06      private int diaAniversario;           // dia de aniversário da caderneta de poupança (entre 1 e 31)
07      private double depositoInicial;       // depósito inicial
08      private double rendimentoAcumulado;   // rendimento acumulado da caderneta de poupança
09
10      // construtor
11      public CadernetaPoupanca(String titular, int diaAniversario, double depositoInicial) {
12          // inicialização de campos de instância com parâmetros
13          this.titular = titular;
14          this.diaAniversario = diaAniversario;
15          this.depositoInicial = depositoInicial;
16
17          // inicialização de rendimentos acumulados
18          this.rendimentoAcumulado = 0;
19      }
20
21      // metódos getter
22      public String getTitular() {
23          return titular;
24      }
25
26      public int getDiaAniversario() {
27          return diaAniversario;
28      }
29
30      public double getDepositoInicial() {
31          return depositoInicial;
32      }
```

```
33
34      public double getRendimentoAcumulado() {
35         return rendimentoAcumulado;
36      }
37
38      // atualização de rendimento acumulado considerando-se saldo atualizado e
39      // taxa percentual de rendimento
40      public void atualizarRendimento(double prTaxa) {
41         // cálculo de rendimento considerando-se taxa de rendimento e rendimento acumulado
42         double rendimento = (depositoInicial + rendimentoAcumulado) * prTaxa / 100;
43
44         // atualização de rendimento acumulado
45         this.rendimentoAcumulado += rendimento;
46      }
47
48      // retorno de saldo atualizado de caderneta considerando-se depósito inicial e rendimento acumulado
49      public double getSaldo() {
50         // retorno de soma de depósito inicial com rendimento acumulado
51         return this.depositoInicial + this.rendimentoAcumulado;
52      }
53
54      // retorno de taxa percentual de rendimento acumulada considerando-se depósito inicial e
55      // rendimento acumulado
56      public double getTaxaRendimentoAcumulada() {
57         return this.rendimentoAcumulado / getSaldo() * 100;
58      }
59
60   }
```

Na forma de campos de instância, os dados encapsulados na classe **`CadernetaPoupanca`** incluem nome do titular, dia de aniversário (um valor entre 1 e 31) e valor de depósito inicial, de modo a serem inicializados na **forma de passagem de parâmetros ao construtor** no momento em que novo objeto da classe for instanciado. Há ainda campo de instância adicional para registro de **rendimento acumulado** da caderneta de poupança. Ademais, somados aos métodos de acesso *getter* de tais campos de instância, são incluídos outros métodos cuja especificação se segue abaixo:

Método	Descrição
`void atualizarRendimento(double prTaxa)`	Atualização de rendimento acumulado considerando-se **saldo atualizado** e taxa percentual de rendimento indicada através do parâmetro **`prTaxa`**
`double getSaldo()`	Retorno de saldo atualizado da caderneta de poupança considerando-se **depósito inicial** e **rendimento acumulado**
`double getTaxaRendimentoAcumulada()`	Retorno de taxa de rendimento acumulada considerando-se valores de **depósito inicial** e de **rendimento acumulado**

Implemente classe utilitária de nome **`CadernetaPoupancaUtil`** para demonstração das capacidades da classe **`CadernetaPoupanca`** da questão anterior. A referida classe deve dispor de método estático **`main`** de modo a permitir que sejam implementadas operações de entrada, pelo usuário, dos dados de 7 (sete) cadernetas de poupança. Sobre estes dados, são eles: nome do titular, dia de aniversário e valor de depósito inicial (a serem encapsulados na forma de objetos da classe **`CadernetaPoupanca`** armazenados em lista implementada através de alguma classe concreta da biblioteca de coleções genéricas da linguagem Java que implemente a interface **`java.util.List<E>`**, a exemplo de **`java.util.ArrayList<E>`** ou **`java.util.LinkedList<E>`**).

2. (Peso: 1,5) Readeque a classe utilitária da **`CadernetaPoupancaUtil`** da questão anterior de tal modo que objetos da classe **`CadernetaPoupanca`** possam ser manipulados após inserção dos mesmos (todos eles) em lista da forma como indicado naquela questão. Deverá ser permitido ao usuário, a qualquer momento, realizar alguma das seguintes operações:
   a) Atualização de rendimento acumulado de alguma das 7 (sete) cadernetas de poupança, cabendo ao usuário, para tal, informar nome do titular e taxa percentual de rendimento (deverá ser exibido saldo da caderneta antes e após aplicação de taxa percentual de rendimento);
   b) Listagem de nomes de titulares e saldos de cadernetas de poupança de determinado dia de aniversário considerando-se que caberá ao usuário previamente informar dia desejado para que sejam selecionadas apenas cadernetas com aniversário naquele dia informado;
   c) Listagem de depósito inicial, dia de aniversário e saldo de cadernetas de poupança de determinado titular considerando-se que caberá ao usuário previamente informar nome do titular para que sejam selecionadas apenas cadernetas em nome daquele titular (para localização de referências de instâncias de cadernetas de poupança armazenadas no *array*, caberá invocação do método **`equalsIgnoreCase`** da classe **`java.lang.String`**);
   d) Encerramento da aplicação.

   Observação: em caso de seleção de alguma das operações elencadas pelos itens *a*, *b* e *c*, após executada, deverá ser permitido ao usuário, a qualquer momento, selecionar nova operação.

3. (Peso: 1,0) Readeque a classe utilitária da **`CadernetaPoupancaUtil`** da questão anterior de tal modo que, quando do encerramento da aplicação, seja exibida quantidade total de "dias de aniversário" entre 1 (um) e 31 (trinta e um) nos quais haja depósito de renumerações provenientes de rendimentos sobre os saldos acumulados de 1 (uma) ou mais cadernetas de poupança entre aquelas cadastradas previamente. Caso hajam 2 (duas) ou mais cadernetas de poupança com um mesmo "dia de aniversário", estas deverão ser contabilizadas apenas uma única vez para se chegar à quantidade total de dias distintos.

Observação: sugere-se aqui uso de alguma implementação de conjunto baseada na interface **`java.util.Set<E>`**, a exemplo de **`java.util.HashSet<E>`**, para fins de contabilização da quantidade total de dias (para tal, considere inclusão dos dias de aniversários das cadernetas de poupança previamente cadastradas em instância de conjunto e, após isso, obtenção de quantidade de elementos inseridos).

4. (Peso: 1,0) Readeque a classe utilitária da **`CadernetaPoupancaUtil`** da questão anterior de tal modo que, quando do encerramento da aplicação, **haja inserção, em uma fila**, de nomes dos titulares de cadernetas de poupança, entre aquelas cadastradas previamente, com inexistência de rendimentos acumulados (ou seja, igual a zero). Após isso, **os nomes inseridos na fila deverão ser desenfileirados e exibidos imediatamente**. Para fins de manipulação de fila, exige-se uso de alguma classe concreta da biblioteca de coleções genéricas da linguagem Java que implemente a interface **`java.util.Queue<E>`**, a exemplo de **`java.util.ArrayDeque<E>`**.

   Observação: a contabilização de quantidade total de "dias de aniversário" distintos da forma como solicitado na questão anterior deve ser mantida.