Caso de estudo – O cartão fidelidade

Cartão de fidelização de clientes das distribuidoras de combustível.

Definição em JAVA da classe `CartaoFidelidade`, que deverá apresentar uma funcionalidade semelhante ao referido cartão.

- Cada cartão deve possuir um titular e um número de pontos (>=0) de bonificação.

Cada **instância** da classe deverá ser capaz de responder adequadamente a um conjunto de mensagens correspondentes às operações de:

- crédito de pontos aquando de um abastecimento ou compras

- débito de pontos (troca de pontos por brindes, se tal for possível)

- consulta do total de pontos do cartão

- consulta do número total de abastecimentos/compras realizados

- consulta do número total de brindes descontados

- apresentação de todos os dados do cartão (cadeia de caracteres)

Comecemos pela definição das variáveis de instância e do (s) método(s) construtor(es):

```
public class CartaoFidelidade {

    //VARIÁVEIS DE INSTÂNCIA:

   private String titular; //nome do possuidor do cartão

   private int pontos, nDebitos, nCreditos;
 //saldo e número de cada tipo de operação
```

```
// CONSTRUTORES:
// o mesmo nome da classe, ausência de um valor de retorno

//Construtor por omissão. O que acontece?

 public CartaoFidelidade() { }


 public CartaoFidelidade (String tit) {
    titular = tit;
    pontos = 0;
    nDebitos = 0;
    nCreditos = 0;
 }
```

```
 // CONSTRUTORES:
// o mesmo nome da classe, ausência de um valor de retorno



public CartaoFidelidade (String tit, int pts) {
    titular = tit;
    pontos = pts;
    nDebitos = 0;
    nCreditos = 0; //Porquê zero e não um? Depende da aplicação...
 }
```

Concentremo-nos agora nos métodos que cada instância da classe CartaoFidelidade terá de ser capaz de proporcionar:
[...]

```
//MÉTODOS DE INSTÂNCIA

//consulta do titular do cartão

  public String getTitular() {
     return titular ;
  }


//modifica o titular

  public void setTitular  ( String titular) {
   this.titular = titular;
}
```

```
//MÉTODOS DE INSTÂNCIA

//consulta do total de pontos do cartão
  public int getPontos() {
     return pontos;
  }


//crédito de pontos aquando de um abastecimento ou compras

  public void creditarPontos(int pontos) {
     this.pontos += pontos;
     nCreditos += 1;
  }
```

```
//MÉTODOS DE INSTÂNCIA

//débito de pontos (troca de pontos por prémios e brindes)

  public void debitarPontos(int pontos) {

  //PRE: pontos <= this.pontos

      this.pontos -= pontos;
      nDebitos += 1;

  }
```

```
//consulta do número total de abastecimentos/compras realizados

 public int getNumCompras() {
     return nCreditos;
  }


//consulta do número total de brindes descontados

  public int getNumTrocas() {
     return nDebitos;
  }
```

```
//apresentação de todos os dados do cartão (cadeia de caracteres)

 public String toString() {

     return  "Cliente: " + titular + "\n" +
             "Saldo do cartão: " + pontos + "\n" +
     "Número de compras: " + nCreditos +
  "\t\tNúmero de brindes: " + nDebitos +"\n";
  }

} // fim da classe
```

```
//apresentação de todos os dados do cartão (cadeia de caracteres)

 public String toString() {

     return  "Cliente: " + titular + "\n" +
             "Saldo do cartão: " + pontos + "\n" +
     "Número de compras: " + nCreditos +
  "\t\tNúmero de brindes: " + nDebitos +"\n";
  }

} // fim da classe
```

Criar um programa para testar a classe agora definida ➔

```
public class TesteCartao {

  public static void main(String[] args) {

    final int ptsCarro = 40, ptsBoneca = 80;

   CartaoFidelidade cardA, cardB, cardC;

   int ptsBrindesPorAtribuir,  nBrindesDistrib, numCompras;

  cardA = new CartaoFidelidade("Adalberto U.M. Fulano");
  cardB = new CartaoFidelidade("Sicrano E. Beltrano", 50);
  cardC = new CartaoFidelidade();
```

```
cardA.creditarPontos(34);
cardA.creditarPontos(44);
cardB.creditarPontos(12);
cardB.creditarPontos(45);

if (cardA.getPontos() >= ptsBoneca)
    cardA.debitarPontos(ptsBoneca);
else
    System.out.println("O seu saldo não permite  esta operação");

cardB.creditarPontos(44);
cardB.creditarPontos(34);
cardB.creditarPontos(44);
```

```
if (cardB.getPontos() >= 3 * ptsCarro){

        cardB.debitarPontos(ptsCarro);
        cardB.debitarPontos(ptsCarro);
        cardB.debitarPontos(ptsCarro);
        //Porquê assim???

 }else

        System.out.println("O seu saldo não permite esta operação!\n");
```

```
    ptsBrindesPorAtribuir = cardA.getPontos() + cardB.getPontos() +
cardC.getPontos();

     nBrindesDistrib = cardA.getNumTrocas() + cardB.getNumTrocas()
+ cardC.getNumTrocas();

     numCompras = cardA.getNumCompras() +
cardB.getNumCompras() + cardC.getNumCompras();

 System.out.println("É necessário provisão para " +
                                        ptsBrindesPorAtribuir + " pontos.");
 System.out.println("Já foram distribuídos  " + nBrindesDistrib +
                                                    " brindes.");
System.out.println("Os nossos clientes já efectuaram  " + numCompras
+ " compras.\n");
```

```
    System.out.println(cardA.toString());

     System.out.println(cardB.toString());

     System.out.println(cardC.toString());
  }
}
```

Resultado da execução: ???

O seu saldo não permite esta operação!

É necessário provisão para 187 pontos.
Já foram distribuídos 3 brindes.
Os nossos clientes já efectuaram 7 operações de compra.

Cliente: Adalberto U.M. Fulano
Saldo do cartão: 78
Número de compras: 2 Número de brindes: 0

Cliente: Sicrano E. Beltrano
Saldo do cartão: 109
Número de compras: 5 Número de brindes: 3

Cliente: **null**
Saldo do cartão: 0
Número de compras: 0 Número de brindes: 0

O seu saldo não permite esta operação!

É necessário provisão para 187 pontos.
Já foram distribuídos 3 brindes.
Os nossos clientes já efectuaram 7 operações de compra.

Cliente: Adalberto U.M. Fulano
Saldo do cartão: 78
Número de compras: 2 Número de brindes: 0

Cliente: Sicrano E. Beltrano
Saldo do cartão: 109
Número de compras: 5 Número de brindes: 3

Cliente: **null**
Saldo do cartão: 0
Número de compras: 0 Número de brindes: 0

Responda se souber:

Como obter o número de cartões emitidos?

Como numerar os cartões emitidos?

[... //classe CartaoFidelidade: alguns métodos adicionais]

```
public boolean equals(Object umObjecto){

  if (umObjecto != null && this.getClass() == umObjecto.getClass() ){
      CartaoFidelidade cf = (CartaoFidelidade) umObjecto;
    return
          titular.equals( cf.titular) &&
         this.pontos == cf.pontos &&  this.nCreditos == cf.nCreditos
         && this.nDebitos == cf.nDebitos;
  }
  else
     return false;
}
```

```
public Object clone(){

  CartaoFidelidade cloneCartao =
                              new CartaoFidelidade(this.titular, this.pontos);

  cloneCartao.nCreditos = this.nCreditos;

  cloneCartao.nDebitos = this.nDebitos;

  return cloneCartao;

}
```

Retomamos o problema do cartão de fidelização e suponhamos que precisamos de um programa para:

- emitir cartões para novos clientes;

- listar os dados de todos os cartões emitidos;

- ordenar os cartões por nome de titular.

```
import java.util.ArrayList;
import myinputs.Ler;
class TesteCartao {
  public static void main(String[] args) {
  ArrayList<CartaoFidelidade>  novosCartoes =
                                        new ArrayList<CartaoFidelidade>( );
    CartaoFidelidade cartao;
    String nome;
    int opcao;
    do{
      System.out.println("\nGasolinho & Gasolinha, Lda.");
      System.out.println("\nMENU DE OPERAÇÕES\n");
       System.out.println("1. Emitir Cartão");
       System.out.println("2. Listar Cartões");
       System.out.println("3. Ordenar cartões por nome de titular");
       System.out.println("0. Sair");
```

```
opcao = Ler.umInt();
 switch(opcao){
        case 1: //***INTRODUZA O SEU CÓDIGO AQUI,
                // antes de ver a solução nas páginas seguintes!***
                break;
        case 2:
                //***INTRODUZA O SEU CÓDIGO AQUI***
                break;
        case 3:
                //***INTRODUZA O SEU CÓDIGO AQUI***
                break;
  }
 }while (opcao!=0) ;
} // fim do main
```

Para nos ajudar na ordenação por selecção vamos um método auxiliar que :
Dada uma ArrayList, **v**,
devolve a **posição** do objecto que contém o titular alfabeticamente **menor** existente na secção limitada por **inicio** e **v.size()-1.**

Nota: O método **int compareTo (String)** devolve um inteiro

=0 se a String que recebe a mensagem é igual à String argumento

<0 se a String que recebe a mensagem é alfabeticamente menor que a String argumento

>0 se a String que recebe a mensagem é alfabeticamente maior que a String argumento

```
private static int procuraMenorAlfab
                              (ArrayList<CartaoFidelidade>  v, int inicio){
   int iMenor = inicio;
   CartaoFidelidade c1, c2;

   for(int i=inicio+1; i < v.size(); ++i){

      c1 =  v.get( i );

      c2 =  v.get( iMenor );

      if (c1.getTitular().compareTo(c2.getTitular())<0)
         iMenor = i;
   }
   return iMenor;
}
```

Uma possível resolução para o problema proposto é então:

```
do{
  [...]

  case 1: //emitir e guardar um novo cartão na ArrayList novosCartoes

     System.out.println("\n\nNome completo titular: ");
     nome = Ler.umaString();
     cartao = new CartaoFidelidade (nome);

     novosCartoes.add (cartao);

     break;
```

```
case 2: //listar o conteúdo do vector novosCartoes

    for(int pos=0; pos < novosCartoes.size(); ++pos){

        cartao = novosCartoes.get(pos);

        System.out.println(cartao.toString());
    }
    break;
```

```
case 3: //ordenar o vector novosCartoes alfabeticamente por nome de
        // titular dos objectos (cartões) guardados

   for(int pos=0; pos < novosCartoes.size(); ++pos){

      //encontrar o menor na secção [pos .. novosCartoes.size()[

       int posMenor =
            procuraMenorAlfab(novosCartoes,pos); //método AUXILIAR
```

```
 // trocar o elemento da posição pos com o elemento da posição
//posMenor, colocando assim o menor elemento na posição pos

      cartao =(CartaoFidelidade)novosCartoes.get(pos);

      novosCartoes.set ( pos,  novosCartoes.get (posMenor) );

      novosCartoes.set ( posMenor ,   cartao  );

    }//for
    break;
  }// switch
 }while (opcao!=0);
[...]
```

✍ Responda se souber:

Como evitar que sejam guardados cartões “repetidos”?

Como eliminar o cartão do “António Silva”?

Como eliminar todos os cartões com nome de titular começado por W?

…

Exercício:

1 – Um produto tem um código, um nome, um preço e a quantidade existente. O código do produto é um inteiro que deve ser atribuído de forma automática cada vez que é criado um novo produto.

a) Para a classe produto defina o cabeçalho, os atributos e um construtor que receba como parâmetro o nome do produto.

b) Construa os getters e setters para os atributos da classe produto.

c) Para a classe Produto construa os métodos toString, equals e clone.

d) Construa uma classe de teste.

2 – Construa um método public e static que receba como parâmetro um objeto do tipo java.util.ArrayList<Produto>. O método deverá devolver o nome do produto mais caro que exista na lista.

3 - Construa um método *public* e *static* que receba como parâmetro um objeto do tipo java.util.ArrayList<Produto> e uma String com o nome de um produto. O método deverá devolver o número de produtos que existem na lista de produtos com nome igual ao nome dado como parâmetro.

Resolução: 1 -

```
public class Produto {
   private static int ultimo=0;
   private int codigo;
   private String nome;
   private double preco;
   private int qtd;

   public Produto(String nome){
      ultimo ++;
      codigo = ultimo;
      this.nome= nome;
      preco =0;
      qtd =0;
   }
```

Resolução:

```
// getters

// setters


public String toString (){
      return "Código:" + codigo + " Nome: " + nome + " Preço: " + preco +
" Existência " + qtd;
    }
```

Resolução:

```
public boolean equals(Object umObjecto){

  if (umObjecto != null && this.getClass() == umObjecto.getClass() ){

        Produto pr = (Produto) umObjecto;
     return
           nome.equals( pr.nome) &&
          this.codigo == pr.codigo &&  this.preco == pr.preco
          && this.qtd == pr.qtd;
  }
  else
     return false;
}
```

Resolução:

```
public Object clone(){

 Produto copia =  new Produto (this.nome);

  copia.codigo = this.codigo;

  copia.preco = this.preco;

  copia.qtd = this.qtd;

  return copia;

}
```

Resolução:

```
// 2 -
   public static String maisCaro (ArrayList<Produto>  lista ){
      String nomeMaisCaro = "";
      double mais = 0.0;
      for (int i=0; i< lista.size(); i++){
         if ( lista.get(i).getPreco() > mais) {
            mais = lista.get(i).getPreco();
            nomeMaisCaro = lista.get(i).getNome();
         }
      }
     return nomeMaisCaro;
    }
```

Resolução:

```
// 3-

 public static int  existem  (ArrayList<Produto>  lista , String n){
       int conta= 0;
       for (int i=0; i< lista.size(); i++){
           if ( lista.get(i).getNome().equals (n)) {
             conta++;
           }
       }
      return conta;
   }
```