1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Programação Orientada a Objetos

Projeto Final

Gestor de compras online

14 Dezembro 2021

Realizado por: André Moreira n 2020239416

Introdução

O objetivo deste projeto consiste em realizar uma aplicação para a venda online de produtos em java. Para realizar esta aplicação existem vários operações que o programa deverá realizar, sendo as mais importantes as seguintes:

1. Cliente poder realizar um login.
2. Cliente poder realizar uma compra.
3. Cliente poder verificar todas as compras realizadas.

Operação 1

A operação 1 consiste, como dito na introdução, em permitir o cliente realizar um **login**. Iniciamos a operação por verificar se o **ficheiro de objeto** já foi criado, cajo ainda não tenha sido criado, lemos os ficheiros de texto para gravar os **clientes**, **produtos** e **promoções** que eles contêm. Cajo tenha sido criado, lemos o ficheiro de objeto para gravar o mesmo.

```
Cliente existe(String texto){
  int encontrado = 0;
  Cliente c = null;

  while(encontrado == 0){

    texto = ler_texto();

    for(Cliente cli : clientes){
      if((cli.getEmail()).toLowerCase().equals(texto.toLowerCase())){
        return cli;
      }
    }
    if(encontrado == 0)
      System.out.println("Nao existe nenhum cliente com esse email.\nQual o seu email?");

  }
  return c;
}
```

Após realizada a leitura dos ficheiros, iniciamos o login do cliente. Começamos por pedir o seu email usando o método **"ler_texto"** e, depois, verificando se é um email existente através do método **"existe"** que se encontra na classe **Supermercados** (classe que contem funções do programa).

```
if(!fileSupermercados.exists()){
  s.ler_ficheiro_produtos(fileProdutos);
  s.ler_ficheiro_promocoes(filePromocoes);
  s.ler_ficheiro_clientes(fileClientes);
  System.out.println("Iniciando login.....");
  System.out.println("Qual o seu email: ");
  cli = s.existe(email);
} else{
  Supermercados s1 = s.ReadObjectToFile(fileSupermercados);
  System.out.println("Iniciando login.....");
  System.out.println("Qual o seu email: ");
  cli = s1.existe(email);
  s = s1;
}
```

Este método verifica o email de todos os clientes, verificando se algum coincide com o **"texto"** (email fornecido pelo cliente), cajo não seja encontrado o email, é pedido de novo um email ao cliente, repetindo o loop ate que seja fornecido um email válido, devolvendo o cliente que contem esse email.

Operação 2

A operação 2 garante a possibilidade do usuário poder realizar uma compra, perguntando ao usuário o produto que ele deseja comprar e verificando se o produto inserido pelo cliente é válido.

E feita uma verificação através de um **loop for**, que verifica todos os nomes dos produtos para ver se algum e equivalente ao fornecido pelo cliente, cajo não seja, avisamos o cliente que não existe o produto que ele forneceu e pedimos outro nome, fazendo isto até que seja fornecido um nome válido.

```
while(indice == -1){

    System.out.println("\nDigite o nome do produto que deseja comprar: ");
    escolha2 = ler_texto();

    for(int x = 0; x < p.size() ; x++){
        if(escolha2.toLowerCase().equals(p.get(x).getNome().toLowerCase()))
            indice = x;
    }

    if(indice == -1)
        System.out.println("Nao exsite nenhum produto com esse nome.");
}
```

Após ser inserido um produto valido iniciamos o processo da compra (Fig.1). Começamos por perguntar ao cliente qual a quantidade do produto deseja comprar, verificando o valor inserido para ver se é um valor válido através do método **"ler_inteiro",** depois de receber um valor válido, verificamos se o produto já foi comprado anteriormente ou se é um produto novo.

Cajo já tenha sido comprado, verificamos se temos **stock** suficiente desse produto e, se tivermos stock suficiente, aumentamos a **quantidade do produto** na compra, o **custo total** e a variável **encontrado** (que nos permite saber que o produto já existia, para não criarmos uma nova compra).

```
void realiza_compra(Carro_compras c){

  if(c.getItem().getStock() >= c.getQuantidade()){
    c.getItem().setStock(c.getItem().getStock() - c.getQuantidade());
    carro.add(c);
    total += c.getItem().getPreco() * c.getQuantidade();
  }
  else{
    System.out.println("Não existe stock sufeciente!");

  }
}
```

Se ainda não tiver sido comprado, criamos uma nova compra, com o produto e a quantidade inserida e realizamos a compra através do método **"realiza_compra"** pertencente à classe **Compra,** este método começa por verificar se o **stock** é suficiente, se sim, retira ao **stock** a **quantidade** comprada, adiciona a **compra** ao **carrinho** e, adiciona ao **total** o **valor da compra**.

```
int menu_compra1(Supermercados s , ArrayList<Produtos> p , Compra compra , int i){
    int encontrado = 0;
    int quantidade;

    do{

    System.out.println("\nQual a quantidade que deseja comprar deste produto?");
    quantidade = ler_inteiro();

    if(quantidade < 0)
        System.out.println("Valor Inserido e Invalido.");

    }while(quantidade < 0);

    for(Carro_compras carrinho : compra.getCarro()){

        if(p.get(i) == carrinho.getItem()){
            if(carrinho.getItem().getStock() < quantidade){
                encontrado++;
                System.out.println("Nao existe stock sufeciente!");
            }else{
             carrinho.setQuantidade(carrinho.getQuantidade() + quantidade);
             carrinho.getItem().setStock(carrinho.getItem().getStock() - quantidade);
             compra.add_total(p.get(i).getPreco() * quantidade);
             encontrado++;
            }
        }
    }
    if(encontrado == 0){
        Carro_compras c1 = new Carro_compras(p.get(i) , quantidade);
        compra.realiza_compra(c1);
    }

return quantidade;
}
```

Fig.1 – Processo da compra

Finalizada a compra do produto, começamos por perguntar ao cliente se ele deseja comprar mais produtos ou não utilizando o método **"fazer_escolha"** para verificar se a escolha encontra-se entre os números apresentados.

```
while(escolha3 != -1){
  System.out.println("Deseja comprar mais produtos?\n\n1->Sim\n2->Nao\n");
  escolha3 = fazer_escolha(3,s);

  if(escolha3 == 1){

    for(Produtos prod: p)
    System.out.printf(prod.toString() + "\n\n");
    escolha3 = -1;
  }

  if(escolha3 == 2){

    System.out.println("Deseja:\n1-> Eliminar produtos do carrinho\n2-> Finalizar compra\n");
    escolha3 = fazer_escolha(3,s);
```

Cajo o cliente escolha comprar mais produtos, mostramos o catálogo de produtos de novo e, realizamos a **compra do produto** novamente. Se escolher que não deseja comprar mais produtos, damos duas opções, **eliminar produtos do carrinho** ou **finalizar a compra**.

```
System.out.println("Deseja:\n1-> Eliminar produtos do carrinho\n2-> Finalizar compra\n");
escolha3 = fazer_escolha(3,s);
```

Se o cliente escolher eliminar um produto (Fig. 2), começamos por perguntar ao cliente se deseja eliminar só um produto ou todos os produtos.

Se o cliente escolher eliminar um produto, começamos por representar o catálogo e perguntar qual o produto que deseja eliminar. Depois de inserido o nome do produto, verificamos se foi inserido uma string (através do método **"ler_texto")**, após a verificação corremos o carrinho de compra para confirmar que o produto esta a ser comprado e, corremos os produtos para obter o índice do produto dentro do **arraylist "Produtos"**, depois de obter o índice, chamamos o método "**eliminar_produto",** que ira eliminar o produto escolhido pelo cliente.

```
void eliminar_produto(Produtos prod , Compra c ){
  int i = 0 ,indice = 0;
  double preco ;

  for(Carro_compras carrinho : c.getCarro()){

    if(carrinho.getItem().equals(prod)){
      indice = i;
      carrinho.getItem().setStock((carrinho.getItem().getStock() + carrinho.getQuantidade()));
      preco = (carrinho.getItem().getPreco() * carrinho.getQuantidade());
      c.setTotal(c.getTotal() - preco);
    }
    i++;
  }

  c.getCarro().remove(indice);
}
```

Neste método, são percorridos todos os **produtos** que o cliente contém no **carrinho**, verificando-se se, o **carrinho** contém o **produto** que o cliente

deseja eliminar. Se o **produto for encontrado**, guardamos o **índice** em que foi encontrado (para depois remover), adicionamos o **stock** que tinha sido retirado com a compra e, removemos o **preço** que o **produto** custava no total. Depois **de acabar o loop for, removemos o produto do carrinho** (importante ser fora do loop pois ocorre erro se for removido um produto da variável que está a ser usada para o for).

```
if(escolha3 == 1){

  System.out.println("1 ->Eliminar um produto\n2 ->Eliminar todos os produtos\n");
  escolha4 = fazer_escolha(3,s);

  if(escolha4 == 1){
    int eliminado = 0;
    System.out.println("Qual produto deseja eliminar?\n");

    for(Carro_compras carrinho : compra.getCarro()){
      System.out.println(carrinho.getItem().toString() + "\n\n");
    }

    System.out.println("\nDigite o nome do produto que deseja eliminar: ");
    escolha2 = ler_texto();

    for(int x = 0; x < p.size() ; x++){
      for(Carro_compras carrinho : compra.getCarro()){
        if(escolha2.toLowerCase().equals(carrinho.getItem().getNome().toLowerCase()))
          if(escolha2.toLowerCase().equals(p.get(x).getNome().toLowerCase())){
            eliminar_produto(p.get(x) , compra);
            eliminado++;
          }
      }
    }
    if(eliminado == 1)
      System.out.println("Produto eliminado com sucesso!");
    else
      System.out.println("Produto nao encontrado!");
  }else{
    compra.getCarro().clear();
  }
}
else{
  fazer_compra = menu_compra2(p , compra , quantidade , s , d , cli);
  escolha3 = -1;
}
}
}
}
break;
```

Fig. 2 – Processo de eliminação de um produto

O outro caso é quando o usuário deseja terminar a compra, nesse caso, chamamos o método **“menu_compra2”** que irá realizar todas as operações necessárias para terminar a compra e, mudamos o valor da variável **fazer_compra** para terminar o loop da compra.

```
int menu_compra2(ArrayList<Produtos> p , Compra compra , int quantidade , Supermercados s , Data d ,
Cliente cli){
  for(Produtos prod : p){
    int encontrado = 0;
    for(Carro_compras carrinho : compra.getCarro()){

      if(prod == carrinho.getItem()){
        quantidade = carrinho.getQuantidade();
        compra.setTotal(compra.getTotal() + prod.Transportar_item());
        encontrado ++;
      }

    }
    for(Promocoes promo: prod.getPromo()){
      if(s.verifica_promo(promo,d) && encontrado != 0){
        compra.setTotal(compra.getTotal() - (promo.promo(compra.getTotal(), quantidade,
prod.getPreco())));
      }
    }
  }

  compra.setTotal(compra.getTotal() + cli.transporte(compra.getTotal()));
  if(!(compra.getCarro().isEmpty()))
    System.out.println("Compra realizada com sucesso!!\nPreco da compra com portes:" +
compra.getTotal() + "€");

  s.add_compra(compra);
  int fazer_compra = -1;
  return fazer_compra;
}
```

A primeira operação realizada no método e um **for loop** de **Produtos** dentro do **ArrayList<Produtos>** , isto será utilizado adicionar o preço do transporte do produto ao total e, para verificar as promoções.

Dentro do **for loop** iremos inicializar uma variável **encontrado** que será utilizada para verificar se o produto se encontra no carrinho, após iniciar a variável, iniciamos outro **for loop** desta vez de **carro de compras** dentro da **compra**, isto permite obter todos os **produtos** que o cliente contem no carrinho de compras, neste segundo **for loop** verificamos se o **prod** e igual ao **produto** que está no carrinho, caso seja, verificamos qual a quantidade desse produto é que o cliente comprou e, gravamos na variável **quantidade**, depois definimos o total como o **total** + **preço do transporte** do produto, que é definido através do método **“Transportar_item”,** e para terminar incrementamos a variável **encontrado.**

```
@Override
public double Transportar_item(){
  return 0.0;
}
```

Alimentares

```
@Override
public double Transportar_item(){
  if(peso > 15.0)
    return 10.0;
  else
    return 0.0;
}
```

Mobilario

```
@Override
public double Transportar_item(){
  return 0.0;
}
```

Limpeza

Após o segundo **for loop** terminar, começamos outro **for loop,** com as **Promoções** que o **produto** contém, dentro deste **for loop** verificamos se o produto tem a promoção ativa na data atual utilizando o método **"verifica_ promo"** e se **encontrado** é diferente de zero, se ambos se verificarem, então definimos o total como o total atual – o valor da promoção, que irá ser determinado pelo método **promo** pertencente a classe **Promocoes.**

```
boolean verifica_promo(Promocoes promo,Data d){

  boolean valido = false;

  if(d.getAno() < promo.getFim().getAno())
    valido = data_verifica(d,promo.getInicio());

  if(d.getMes() < promo.getFim().getAno() && d.getAno() == promo.getFim().getAno())
    valido = true;

  else if(d.getMes() == promo.getFim().getMes() && d.getAno() == promo.getFim().getAno() && d.getDia()
<= promo.getFim().getDia())
    valido = true;

  return valido;
}
```

```
@Override
public double promo(double total, int quantidade,double preco){
  double promo = 1;
  double valor_retirado = 0;

  for(int i = 0 ; i < quantidade; i++){

    if(promo > 0.5){
      valor_retirado += preco * (1 - promo);
      promo = promo - 0.05;
    }
    else{
      total = total * 0.5;
      return total;
    }
  }

  return valor_retirado;

}
```

Pague_menos

```
@Override
public double promo(double total,int quantidade, double preco){

  double preco_descontado = 0;

  while(quantidade - 4 >= 0){
    quantidade = quantidade - 4;
    preco_descontado += preco;
  }

  return preco_descontado;
}
```

Pague_3_leve_4

Após o **for loop** terminar adicionamos ao total o transporte para o cliente que será calculado utilizando o método **"transporte"** da classe **cliente,** dizemos ao cliente que a compra foi realizada com sucesso, mostrando o preço total da compra, adicionamos a compra ao **<ArrayList>Compras** e, para terminar, damos valor de -1 á variável **fazer_compra** que vai ser usada no return para depois terminar o loop das compras.

Operação 3

A operação 3 garante permite ao usuário verificar todas as compras feitas na data atual. Para realizar esta operação necessitamos apenas de um método, **"mostrar_compras".**

```
void mostrar_compras(Data d){
  int check_compra = 0;

  for(Compra c: compras){
    if(data_verifica(d , c.getData()) && !(c.getCarro().isEmpty())){
      System.out.println(c.toString());
      check_compra++;
    }
  }

  if(check_compra == 0)
    System.out.println("Ainda nao foram realizadas compras.");
}
```

Este método começa por fazer um **for loop** de compras dentro do **ArrayList <Compras>**, verificamos se a compra não foi feita depois da data atual com o método **"data_verifica"** e verificamos se o carrinho não esta vazio, depois de fazer a verificação, apresentamos ao cliente a compra e, incrementamos a variável **check_compra**. Após o **for loop** terminar, verificamos se a variável **check_compra** tem o valor 0, se sim, dizemos ao cliente que ainda não foram feitas compras.

```
boolean data_verifica(Data d, Data d1){
  boolean return_statement = false;

  if(d.getAno() > d1.getAno())
    return_statement = true;
  else if(d.getAno() == d1.getAno() && d.getMes() > d1.getMes())
    return_statement = true;
  else if(d.getAno() == d1.getAno() && d.getMes() == d1.getMes() && d.getDia() >= d1.getDia())
    return_statement = true;

  return return_statement;
}
```

Muda Data

```
case(3):
//pedir e verificar a data ate que uma data correta seja fornecida.
do{
  dias = s.le_data();
}while(d.verifica_data(Integer.parseInt(dias[0]),Integer.parseInt(dias[1]),Integer.parseInt(dias[2]))
== false);

d = new Data(Integer.parseInt(dias[0]),Integer.parseInt(dias[1]),Integer.parseInt(dias[2])); //Tornar
valores fornecidos pela data em inteiros.
System.out.println(d.toString() + "\n");

break;
```

Outra operação que foi adiciona para permitir testar melhor o programa foi poder mudar a data, esta operação pede uma data ao usuário e verifica se a data é correta utilizando o método **“le_data” *,** depois de verificada, torna a string dada pelo usuário em int e muda a data para a data fornecida.

*(nota, métodos le_data,le_int,le_string serão apresentados no fim)

Ficheiros

```java
public Supermercados ReadObjectToFile(File f) {
  try {

    FileInputStream fileIn = new FileInputStream(f);
    ObjectInputStream objectIn = new ObjectInputStream(fileIn);
    Supermercados s = (Supermercados) objectIn.readObject();
    objectIn.close();
    System.out.println("Objeto foi lido do ficheiro.");
      return s;

  } catch (FileNotFoundException | ClassNotFoundException ex) {
    ex.printStackTrace();
  }catch(IOException ex){
    System.out.println("Erro a escrever para o ficheiro.");
  }
  return null;
}
```

Este método serve para ler os objetos de um ficheiro, neste caso, guardando os objetos do tipo **Supermercados** na variável **s**.

```java
public void ler_ficheiro_clientes(File myFil){
  try {
    Scanner myReader = new Scanner(myFil);
    while (myReader.hasNextLine()) {
      String data = myReader.nextLine();
      String[] quebra = data.split(",");
      Data d = new Data(Integer.parseInt(quebra[4]), Integer.parseInt(quebra[5]),
Integer.parseInt(quebra[6]));
      if(Objects.equals(quebra[7], "frequente")){
        clientes.add(new Frequente(quebra[0], quebra[1], quebra[2], Integer.parseInt(quebra[3]),d));
      } else{
        clientes.add(new Normal(quebra[0], quebra[1], quebra[2], Integer.parseInt(quebra[3]),d));
      }
    }
    myReader.close();
  } catch (FileNotFoundException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  } catch(NumberFormatException e) {
    System.out.println("Erro a converter texto em inteiro.");
  }
}
```

Este método serve para ler os ficheiros de clientes, começamos por fazer um **loop** enquanto existir linhas, logo asseguir, damos **.split(,)** e guardamos todos os valores no **array quebra**, depois, definimos a **data** como a data recebida pelo ficheiro, e, verificamos se o cliente é **frequente** ou **regular**, apos a verificação, criamos o cliente com os dados que tem dentro do ficheiro.

```
public void ler_ficheiro_produtos(File myFil){
  try {
    Scanner myReader = new Scanner(myFil);
    while (myReader.hasNextLine()) {
      String data = myReader.nextLine();
      String[] quebra = data.split(",");
      if(Objects.equals(quebra[0], "alimentares")){
        p.add(new Alimentares(Integer.parseInt(quebra[1]), Integer.parseInt(quebra[2]), quebra[3],
quebra[4], Double.parseDouble(quebra[5]), Integer.parseInt(quebra[6])));
      } else if (Objects.equals(quebra[0], "mobiliario")){
        p.add(new Mobiliario(Double.parseDouble(quebra[1]), Double.parseDouble(quebra[2]), quebra[3],
quebra[4], Double.parseDouble(quebra[5]), Integer.parseInt(quebra[6])));
      }else{
        p.add(new Limpeza(Integer.parseInt(quebra[1]), quebra[2], quebra[3],
Double.parseDouble(quebra[4]), Integer.parseInt(quebra[5])));
      }
    }
    myReader.close();
  } catch (FileNotFoundException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  } catch(NumberFormatException e) {
    System.out.println("Erro a converter texto em inteiro.");
  }
}
```

Com este método é nos permitido ler ficheiros de produtos, começamos por, como no método anterior, por fazer um **loop** enquanto existir linhas, depois, realizamos um split(,) gravando no **array quebra**, depois verificamos que tipo de produto é que temos e, criamos um novo produto com os dados que temos.

```
public void ler_ficheiro_promocoes(File myFil){
  try {
    Scanner myReader = new Scanner(myFil);
    while (myReader.hasNextLine()) {
      String data = myReader.nextLine();
      String[] quebra = data.split(",");
      Data d = new Data(Integer.parseInt(quebra[0]), Integer.parseInt(quebra[1]),
Integer.parseInt(quebra[2]));
      Data d1 = new Data(Integer.parseInt(quebra[3]), Integer.parseInt(quebra[4]),
Integer.parseInt(quebra[5]));
      if(Objects.equals(quebra[6], "pague_3_leve_4")){
        for (int i = 0; i < p.size(); i++){
          if(p.get(i).getNome().equals(quebra[7])){
            p.get(i).add_promo(new Pague_3_leve_4(d, d1));
          }
        }
      } else{
        for (int i = 0; i < p.size(); i++){
          if(p.get(i).getNome().equals(quebra[7])){
            p.get(i).add_promo(new Pague_menos(d, d1));
          }
        }
      }
    }
    myReader.close();
  } catch (FileNotFoundException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
  } catch(NumberFormatException e) {
    System.out.println("Erro a converter texto em inteiro.");
  }
}
```

Este método inicia como os anteriores, um loop enquanto existir linhas e dando os valores do split ao array quebra, logo asseguir a fazer isso, começamos por definir a data de início e fim da promoção com os valores obtidos no ficheiro, depois, verificamos que tipo de promoção é, depois de verificar, vemos qual o produto que tem esta promoção e, damos add a promoção no produto.

Métodos de Leitura

```java
String ler_texto(){

  String str;

  try{

    Scanner sc = new Scanner(System.in);
    str = sc.nextLine();

  }
  //Se o valor for um valor causar um erro, ira ser avisado ao usuario que o valor nao e valido.
  catch (java.util.InputMismatchException e){
    System.out.printf("Valor Introduzido nao e valido.");
    return null;
  }

  return str;
}
```

Este método é utilizado para ler uma **string**, o método faz a verificação se o valor dado pelo utilizador é uma **string**, cajo seja, devolve a **string**, se não, dizemos ao utilizador que o valor não e valido e devolvemos **null**.

```java
String[] le_data(){

  int erro;
  String data;
  String[] dias;

  do{

    erro = 0;
    System.out.printf("Data:(dd/mm/aa) ");
    data = ler_texto();
    dias = data.split("/"); //Permite separar a data para obter [DIA,MES,ANO]

    if(dias.length == 0){
      System.out.println("Data fornecida deve ser do tipo DIA/MES/ANO...");
      data = null;
      Arrays.fill(dias,null);
      erro++;
    }
    else if(erro == 0 && (dias[0].equals(data) || dias.length != 3)  ){ //Verifica se a data fornecida
é do tipo DIA/MES/ANO
      System.out.println("Data fornecida deve ser do tipo DIA/MES/ANO...");
      data = null;
      Arrays.fill(dias,null);
      erro++;
    }

    if(erro == 0){
      try{
        Integer.parseInt(dias[0]); //Verificar se o valor que foi fornecido e possivel converter para
um inteiro.
        Integer.parseInt(dias[1]);
        Integer.parseInt(dias[2]);
      }
      catch (java.lang.NumberFormatException e){
        data = null;
        System.out.println("A data fornecida nao e valida!"); //Nao e possivel converter para inteiro,
e avisado o usuario.
      }
    }

  }while(data == null);

  return dias;

}
```

Método que nos permite ler uma data, começamos por fazer um **loop** até termos uma **data válida**, pedimos ao utilizador a data, após ler o texto recebido

fazemos split(/) para separar o dia,mês e ano, logo asseguir verificamos se só foram escritas barras, ou seja, o array irá estar vazio, cajo esteja, dizemos que deu erro, depois, verificamos se o tamanho é diferente de 3 ou, se o dias[0] é equivalente á string, ou seja, não foram inseridas barras, se sim, avisamos que deu erro outra vez, no final, verificamos se os valores recebidos podem ser tornados em inteiros, cajo não possam, dizemos ao usuário que é inválido

```
int ler_inteiro(){

  int n;
  System.out.printf("Digite o numero:");
  try{

    Scanner sc = new Scanner(System.in);
    n = sc.nextInt();

  }
  //Se o valor for um valor causar um erro, ira ser avisado ao usuario que o valor nao e valido.
  catch (java.util.InputMismatchException e){
    return -1;
  }

  return n;
}
```

Este método permite-nos ler inteiros, pedindo ao usuário um numero, depois de inserido verificamos se é inteiro, cajo seja devolvemos o valor, cajo contrário devolvemos -1.