Programação Orientada aos Objetos 2020

App TrazAqui!

André Sousa(a87999), Armando Silva(a87949)

11 de Junho de 2020

Licenciatura em Ciências da Computação Universidade do Minho

Conteúdo:

1. Introdução

Este relatório irá abordar a realização do projeto realizado no âmbito da Unidade Curricular **Programação Orientada aos Objetos**, que consiste na criação de uma app("TrazAqui") de encomendas. Este trabalho foi realizado tendo como base todos os conteúdos lecionados nas aulas.

2. Usuários TrazAqui

Todos os usuários da app "TrazAqui" terão de iniciar sessão via credenciais(usuário/password), e depois dentro da app poderão fazer pedidos de encomendas, classificar entregas e verificar os seus pedidos de encomendas.

2.1 **Adicionar pedido de encomenda**

Dentro da app os Usuários poderão fazer um pedido de encomenda, que basicamente é adicionar uma encomenda a um Map<String, Encomenda> pedidosEncomendas , que guarda todos os pedidos de encomendas de todos os usuários , sujeitos a aceitação.

2.2 **Verificar pedidos de encomenda**

A qualquer momento os usuários poderão verificar se a sua encomenda esta pendente a aceitação, através de um:

System.out.println(app.getPedidosEncomendas);

2.3 **Classificar Voluntários/Transportadora**

Os usuários poderão também, no final de encomenda, classificar o voluntário e transportadora encarregues da entrega da encomenda dando-lhes uma classificação de 0-5, que se juntara a media de classificações dadas por todos os usuários a certo voluntario/transportadora

3 Voluntários

Os voluntários da app "TrazAqui" , após iniciarem sessão poderão verificar pedidos de encomenda , Registar tempo de encomenda , Aceitar fazer encomenda;

3.1 **Verificar pedidos de encomenda**

A qualquer momento os voluntários poderão verificar as encomendas pendentes a aceitação, através de um:

system.out.println(app.getPedidosEncomendas);

4 Transportadoras

As trasnsportadoras da app “TrazAqui” poderão, após realizarem o login, verificar os pedidos de encomenda, verificar o total faturado a qualquer momento.

4.1- Verificar pedidos de encomenda

A qualquer momento as transportadoras poderão verificar as encomendas pendentes a aceitação, através de um:

```
System.out.println(app.getPedidosEncomendas);
```

4.2-Verificar total faturado

A qualquer momento as transportadoras poderão verificar quanto é que faturaram ate ao momento em encomendas realizas verificando e somando o custo de todas as encomendas guardadas em:

app.getEncConcluidas() -> Map que guarda todas as encomendas concluídas e todas as informações associadas a encomenda, incluído o custo,custo este que e calculado na GestaoEncomendas através da função :

```
public double custoTransportadora(Loja l, Utilizador u, Transportadora t, Encomenda e)
```

5 Loja

As lojas da app “TrazAqui” poderão verificar a qualquer momento todos os pedidos que lhe foram efetuados e também todas as encomendas que já foram concluídas com sucesso.

5.1- Verificar pedidos de encomenda

A qualquer momento as lojas poderão verificar as encomendas pendentes a aceitação, através de um:

```
System.out.println(app.getPedidosEncomendas());
```

5.2-Verificar encomendas concluídas

A qualquer momento as lojas poderão verificar as encomendas concluídas , através de um:

System.out.println(app.getEncConcluidas());

6 Utilizadores app “TrazAqui”

Todos os utilizadores da app “TrazAqui” poderão se registar (usuário,loja,voluntario,transportadora) de forma gratuita , e cada um dos usuários terá acesso a um menu especifico para cada tipo de utilizador, poderão também verificar qual é o top 10 usuários que mais pedidos fazem na loja.

6.1-Login

O login é feito a partir de credenciais (usuário/password) que são guardados num map do tipo Map<String, Utilizador> getAllUtilizadores (); e são verificados através destas funções :

```
public boolean validaUtilizador(String CodU, String pass){
    return this.AllUtilizadores.containsKey(CodU) && this.AllUtilizadores.get(CodU).getPwU().equals(pass);
}
```

6.2-Registar

Os utilizadores da app “TrazAqui” ao fazer o seu registo na app serão questionados se se querem registar como: usuário, loja, transportadora ou Voluntário. E serão pedidas algumas informações consoante o tipo de usuário a ser registado, todos estes resgistos serão guardados em maps do género:

```
public Map<String, Transportadora> getAllTransportadoras(){
    Map<String, Transportadora> t = new HashMap<>();
    for(Map.Entry<String, Transportadora> e : this.AllTransportadoras.entrySet()){
        t.put(e.getKey(), e.getValue().clone());
    }
    return t;
}
```

Através de funções do género:

```
public void nregistarTransportadora(String CodEmpresaT, String NomeT, GPS gpsT, String NifT, double RaioT, double precokmT, String pwT) throws JaRegistadoException{
    Transportadora t = new Transportadora(CodEmpresaT, NomeT, gpsT, NifT, RaioT, precokmT, pwT, 0, 0,0);
    if (this.AllTransportadoras.containsKey(t.getCodEmpresa())) throw new JaRegistadoException("Esta Transportadora ja foi registada! : " + t.getCodEmpresa());
    else this.AllTransportadoras.put(CodEmpresaT, t.clone());
}
```

6.2-Verificar top 10 usuários que mais usam a app

Os utilizadores da app “TrazAqui” no menu inicial terão acesso ao “Top 10 usuários” que lhes mostrará ,em formato de lista ordenada, os usuários que mais pedidos de encomenda efetuam na nossa app , acompanhado do numero de pedidos efetuados ate ao momento, o top 10 é implementado através de uma função que verifica as 10 palavras(códigos de usuários) que aparecem com mais frequência nos pedidos de encomenda através desta função :

```
//Top10Utilizadores
public static Map<String, Integer> getWordFrequencies(List<String> words) {
    Map<String, Integer> wordFrequencies = new LinkedHashMap<String, Integer>();
    if (words != null) {
        for (String word : words) {
            if (word != null) {
                word = word.trim();
                if (!wordFrequencies.containsKey(word)) {
                    wordFrequencies.put(word, 0);
                }
                int count = wordFrequencies.get(word);
                wordFrequencies.put(word, ++count);
            }
        }
    }
    return wordFrequencies;
}
```

6.3-Verificar top 10 transportadoras

Os utilizadores da app “TrazAqui” no menu inicial terão acesso ao “Top 10 Transportadoras” que é feito baseado no número de quilómetros percorridos por uma transportadora até ao momento, este “top 10 ” é implementado através de um Map<String,Double> buscatransportadoras(ArrayList<Double> lista); que adiciona ao map a correspondência entre usuário e numero total de km percorridos por ordem decrescente ,através das funções :

```
public ArrayList<Double> sortlista(ArrayList<Double> list){
    Collections.sort(list);
    Collections.reverse(list);
    System.out.println(list+"\n");
    return list;
 }
public ArrayList<Double> so10(ArrayList<Double> list){
    ArrayList<Double> listnova=new ArrayList<>();
   for(int i = 0;i<11;i++){
      listnova.add(list.get(i));
     }
   System.out.println(listnova+"\n");
   return listnova;
 }
public Map<String,Double> buscatransportadoras(ArrayList<Double> lista){
    Map<String,Double> listanova = new HashMap<>();
    for(int i=0;i<lista.size();i++){
       for(Map.Entry<String,Transportadora> t : this.AllTransportadoras.entrySet()){
            if(lista.get(i)== t.getValue().getKm()){
              listanova.put(t.getKey(),t.getValue().getKm());
            }
      }
    }

    return listanova;
 }
```

A ArrayList<Double> sortlista(ArrayList<Double> list) que ordena a lista de todas os km de todas as transportadoras por ordem decrescente , a ArrayList<Double> so10(ArrayList<Double> list) que limita o array ordenado apenas a exatamente 10 elementos para o “top 10”, e Map<String,Double> buscatransportadoras(ArrayList<Double> lista) que associa o numero de quilómetros ao código da transportadora.

6.4-Read dos logs

Toda a informação da aplicação “TrazAqui” é guardada em logs que poderá ser carregada ou não pelos utilizadores (escolha pessoal), esta informação e guardada em ficheiros de texto , e é lido a partir de funções do género :

```
public static void parseTransportadora(String transportadora, Login app) throws JaRegistadoException {
    String[] campos = transportadora.split(",");
    String codT = campos[0];
    String nome = campos[1];
    GPS gps = new GPS(Double.valueOf(campos[2]), Double.valueOf(campos[3]));
    String nif = campos[4];
    double raio = Double.valueOf(campos[5]);
    double preco = Double.valueOf(campos[6]);
    String pass = campos[7];
    int classi = Integer.parseInt(campos[8]);
    double fat = Double.valueOf(campos[9]);
    double km = Double.valueOf(campos[10]);
    app.readregistarTransportadora(codT, nome, gps, nif, raio, preco, pass,classi, fat,km);
}
```

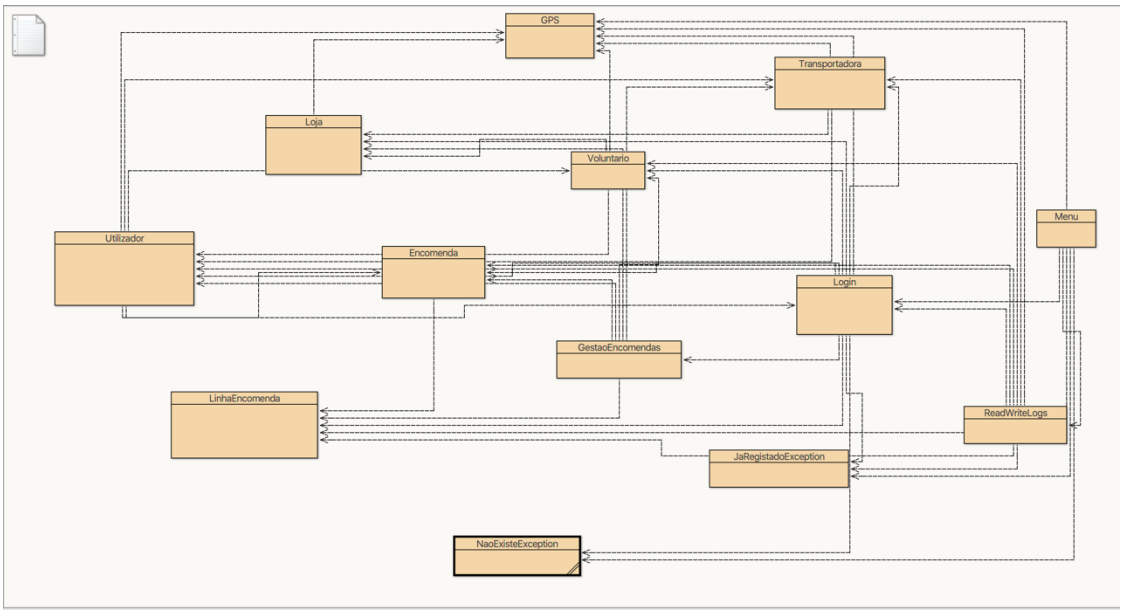
E

```
public void readregistarTransportadora(String CodEmpresaT, String NomeT, GPS gpsT, String NifT, double RaioT, double precokmT, String pwT, int classi, double fat,doub
    Transportadora t = new Transportadora(CodEmpresaT, NomeT, gpsT, NifT, RaioT, precokmT, pwT, classi, fat,km);
    if (this.AllTransportadoras.containsKey(t.getCodEmpresa())) throw new JaRegistadoException("Esta Transportadora ja foi registada! : " + t.getCodEmpresa());
    else this.AllTransportadoras.put(CodEmpresaT, t.clone());
}
```

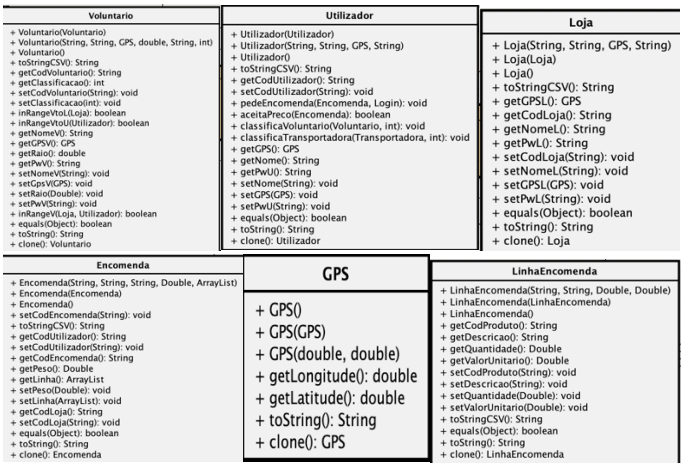
A parte da escrita nos ficheiros e feita através de funções do género:

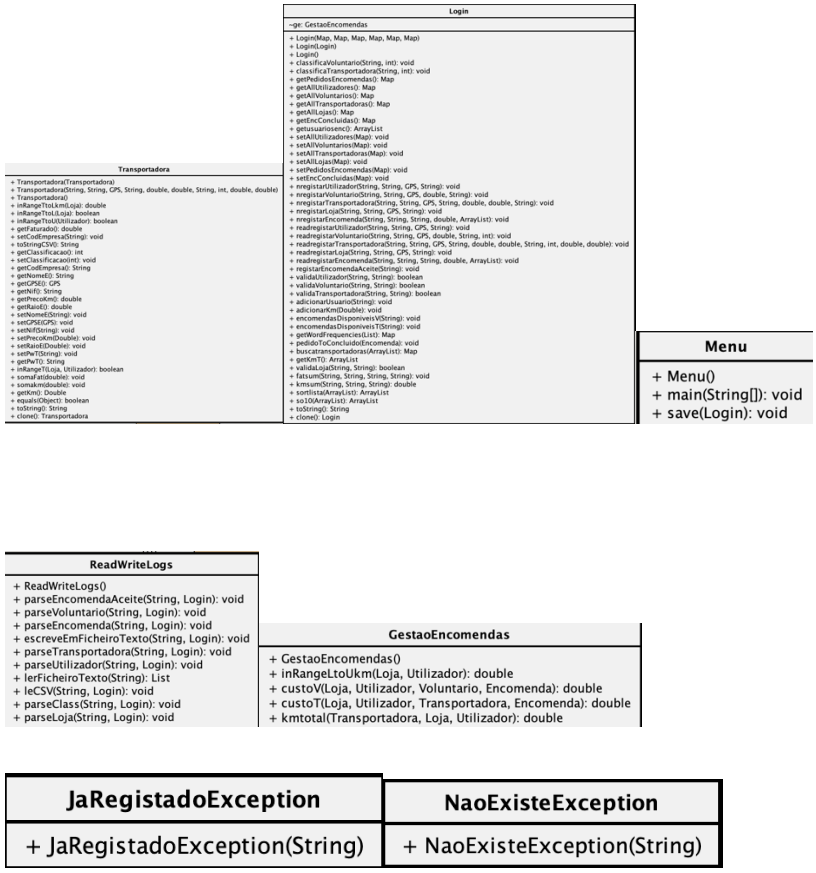
```
public static void escreveEmFicheiroTexto(String nomeFicheiro, Login app) throws FileNotFoundException{
    PrintWriter fich = new PrintWriter(nomeFicheiro);
    for(Utilizador u: app.getAllUtilizadores().values()) {
        fich.println(u.toStringCSV());
    }
    for(Voluntario v: app.getAllVoluntarios().values()){
        fich.println(v.toStringCSV());
    }
    for(Transportadora t: app.getAllTransportadoras().values()){
        fich.println(t.toStringCSV());
    }
    for(Encomenda e: app.getPedidosEncomendas().values())
        fich.println(e.toStringCSV());

    for (Encomenda e: app.getEncConcluidas().values())
    fich.println("Aceite:" + e.getCodEncomenda());

    fich.flush();
    fich.close();
}
```

7 Diagrama de classes



7.1 Métodos:





(Obs:. Tivemos dificuldades a criar o diagrama de classes, e tivemos de improvisar.)

8 Conclusão

Em suma, apos a conclusão deste trabalho o grupo considera que fez um bom trabalho, visto que passamos pelos tópicos pedidos, e concluímos que a realização deste trabalho nos foi bastante útil para compreender melhor o funcionamento da linguagem java num contexto mais prático e nos ajudou a interiorizar melhor a matéria dada nas aulas teóricas.