MJV

School de Java
GRUPO 4

Power Classes

O nosso desafio é explorar as principais classes da linguagem Java.

Requisitos

1. Realizar uma breve descrição da classe em questão;
2. Apresentar alternativas de instanciação de objetos com Construtor ou metódos Singleton;
3. Apresentar no mínimo 04 métodos mais utilizados destacando o seu contrato (tipo retorno + nome + parâmetros);
4. Apresentar se alguns dos métodos é sobrecarregado;
5. Realizar uma demonstração adaptando o uso dos métodos (mínimo 04) em situações do cotidiano (use a imaginação).

Classes - Grupo 04

- java.util.Formatter
- java.time.format.DateTimeFormatter
- java.text.DecimalFormat

MEMBROS

Evelyn Carolina
Dominic L. Barbosa
Joana Silva
Lucca Bugatti
Mayara Saavedra
Patricia Ferreira de Sousa
Tatiane Pimenta Leal
Wagner dos Santos

Preparação para uso, aplicação e demonstração das classes

Classes Formatter, DateTimeFormatter e DecimalFormat

Essas classes são utilizadas para formatar impressões, ou seja, às saídas para dados numéricos de string e de data/hora (no caso do Formatter e DateTimeFormatter) e números reais (para DecimalFormat).

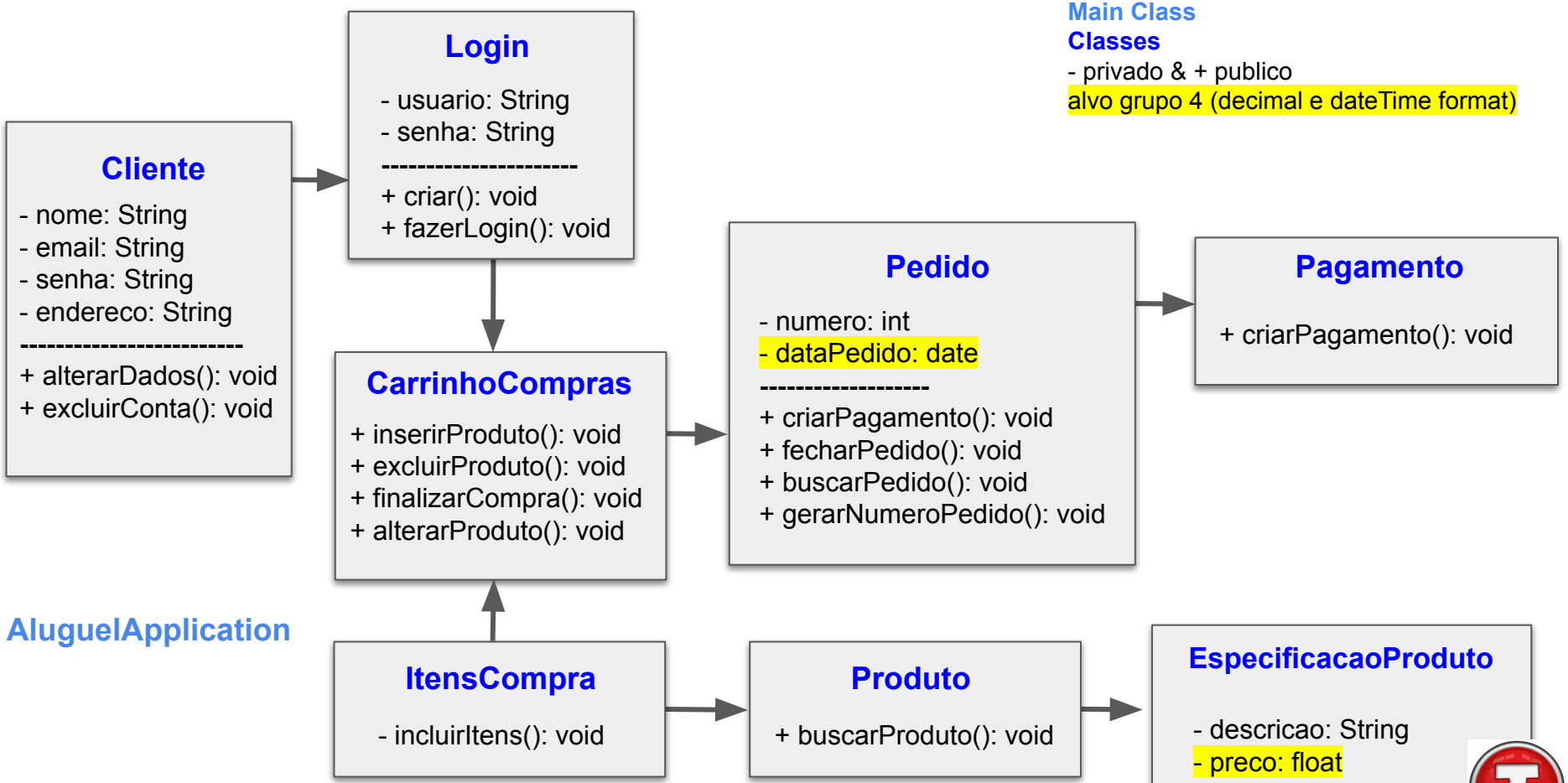
Com o uso deles, é possível formatar saídas de acordo com a necessidade do programador, levando também em consideração e de forma automática a localidade aplicável com o uso do “Locale” e “LocalDate” para o Formatter e DateTime, respectivamente.

A documentação oficial traz ainda formatadores tabelados pré-definidos, com formatação ISO quando necessário.

Informações mais detalhadas e exemplos além do projeto do grupo 4 encontram-se em nosso README.

Diagrama de Classes “Aluguel de Carros” versão inicial

Legenda
Main Class
Classes
- privado & + publico
alvo grupo 4 (decimal e dateTime format)



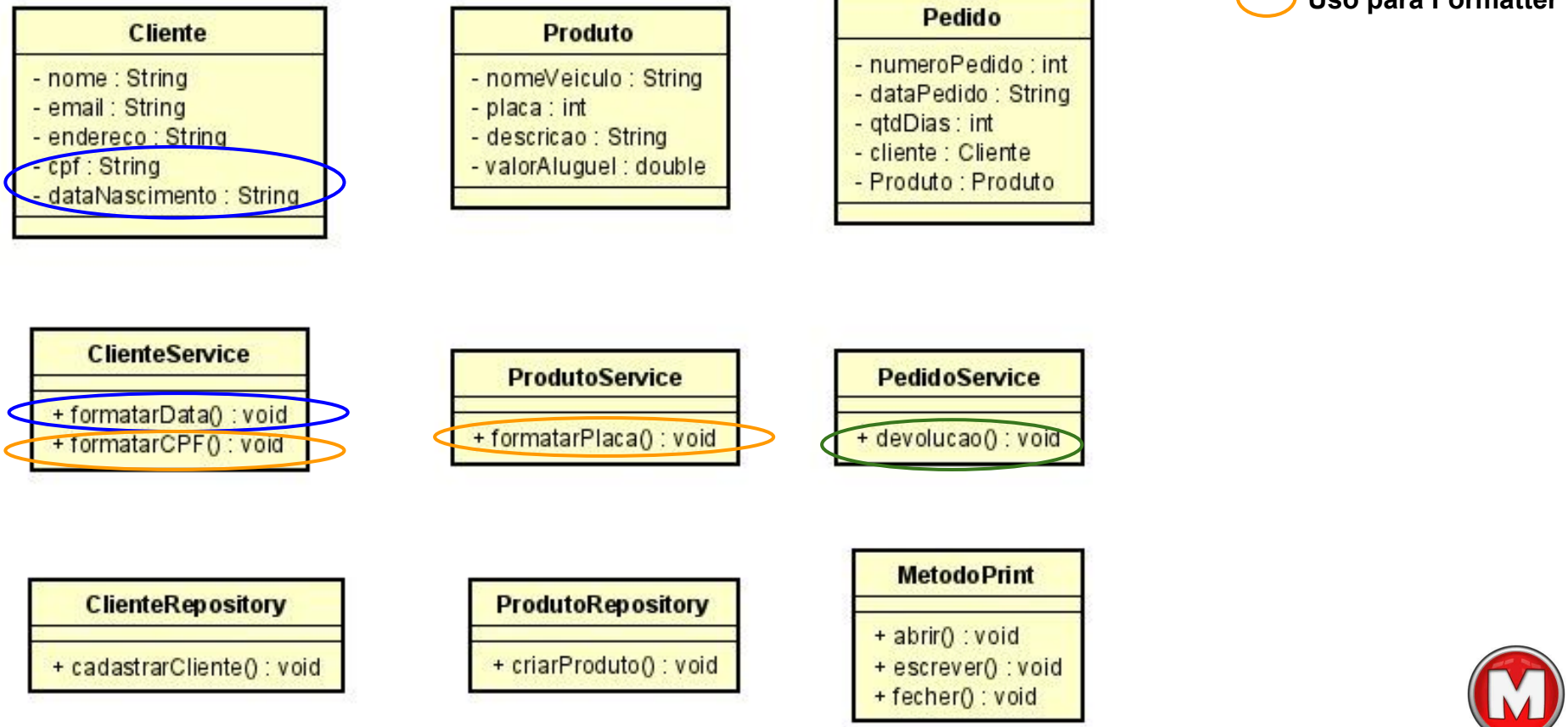
Referências: https://java-programming.mooc.fi/part-11/1-class-diagrams / https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml

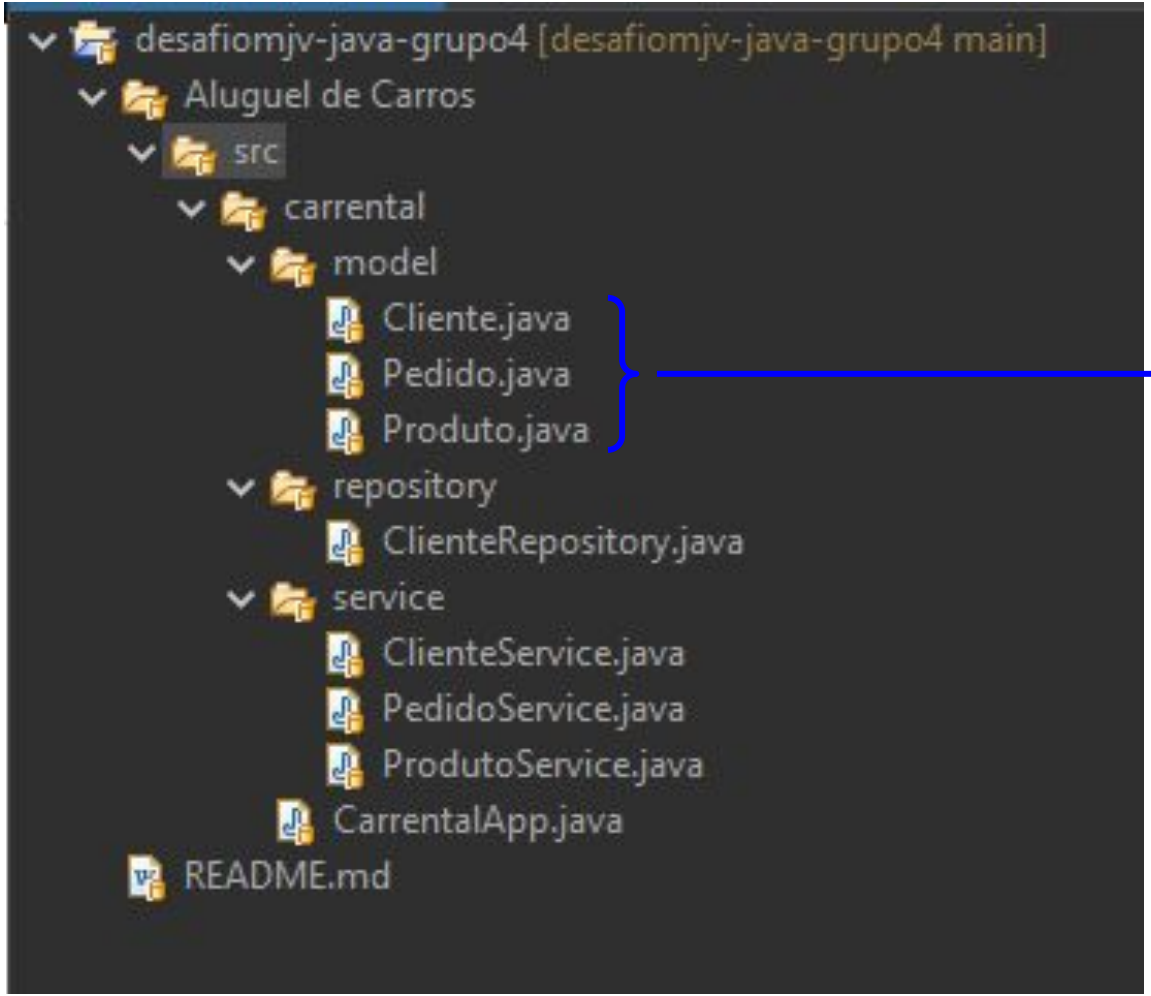
Simplificação da versão inicial do Diagrama de Classes

Uso para DateTimeFormatter

Uso para DecimalFormat

Uso para Formatter





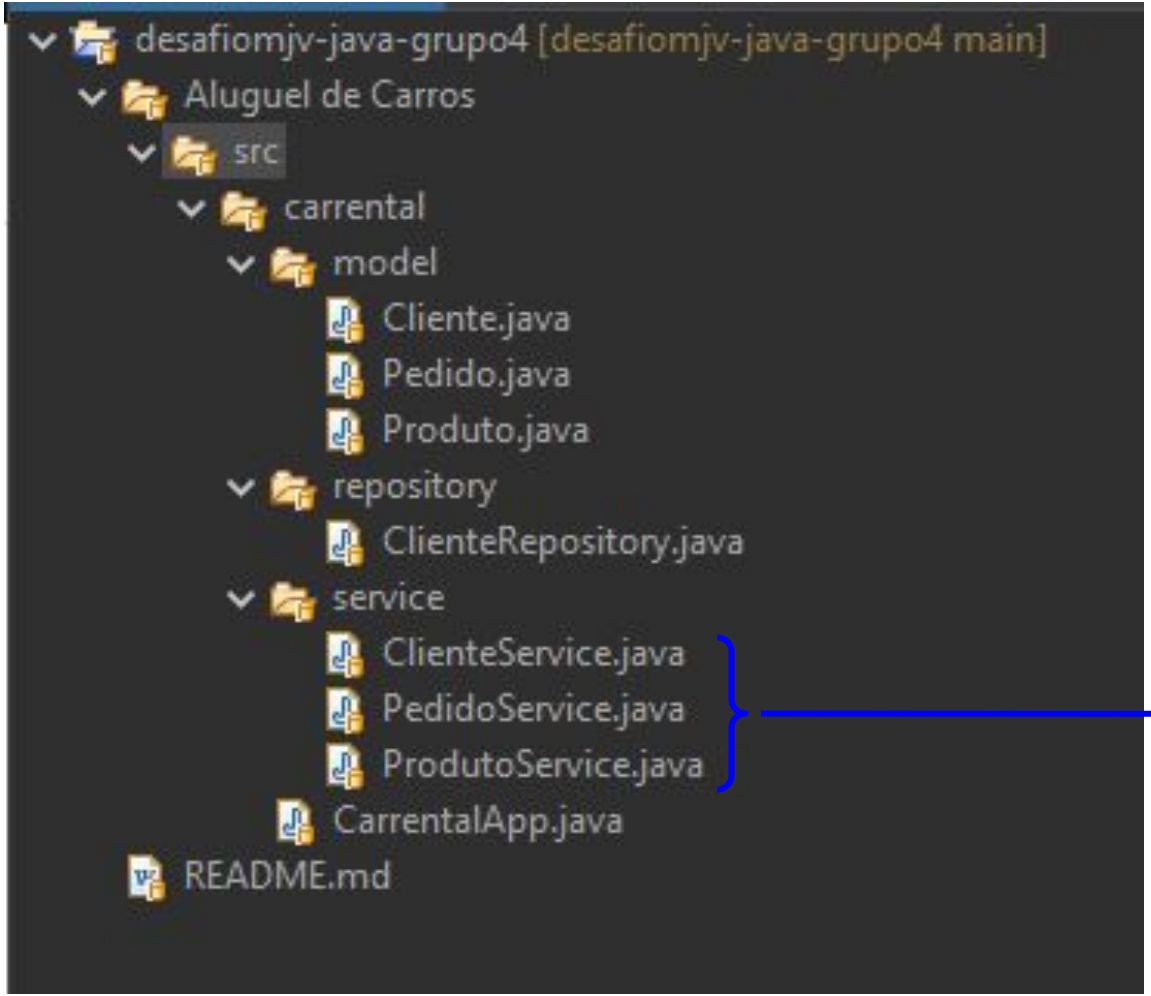
Construtores, Getters & Setters para a criação dos objetos

```java
package carrental.model;

import java.util.Date;

import carrental.service.ClienteService;

public class Cliente {
	private String nome;
	private String email;
	private String senha;
	private String endereco;
	private String dataNascimento;
	private String cpf;

	public Cliente(String nome, String email, String senha, String endereco, Date dataNascimento, String cpf) {
		ClienteService service = new ClienteService();
		this.nome = nome;
		this.email = email;
		this.senha = senha;
		this.endereco = endereco;
		this.dataNascimento = service.formatarData(dataNascimento);
		this.cpf = cpf;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
```

Visual da aplicação dos construtores na classe “Cliente”

```java
@Override
public String toString() {
    DecimalFormat valorDecimal = new DecimalFormat("R$ ##.00");
    Locale localBR = new Locale("pt","BR");
    NumberFormat moedaBR = NumberFormat.getCurrencyInstance(localBR);



    return String.format("Detalhes do Produto "
            + "\nNome Veiculo: " + nomeVeiculo
            + "\nPlaca: " + placa
            + "\nDescricao: " + descricao
            + "\nQuantidade de dias que ficará alugado: " + qtdDias
            + "\nValor do Aluguel: " +  valorDecimal.format(valorAluguel)
            + "\nValor do Total: " +  moedaBR.format(valorTotal));

}
```

Para a DecimalFormat a grande questão era fazer com que os detalhes do produto estivessem alinhados com os valores atrelados a ele (valor do aluguel e valor total pelos dias locados)



No Service Package incluímos os métodos para a criação das ações do sistema, incluindo a formatação de data de nascimento e CPF do Cliente

```java
package carrental.service;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ClienteService {



        public Date formatarData(String dataNasciemnto) throws ParseException {

                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

                Date dataFormatada = formatter.parse(dataNasciemnto);
        return dataFormatada;
        }

        public String formatarCPF(String cpf) {

                return(cpf.substring(0, 3) + "." + cpf.substring(3, 6) + "." +
                                cpf.substring(6, 9) + "-" + cpf.substring(9, 11));
        }

```

Utilizamos às classes Date, ParseException e SimpleDateFormat, para a formatação da data de nascimento, todas da “família de formatação” da documentação oficial do Java.

Em um dos testes, encontramos dificuldades e descobrimos no Sysout que o Java exige número de 1-12 para mês e também que seja separado por barra / por isso tivemos que contornar o problema.

```
21          public String formatarCPF(String cpf) {
22
23                  return(cpf.substring(0, 3) + "." + cpf.substring(3, 6) + "." +
24                                  cpf.substring(6, 9) + "-" + cpf.substring(9, 11));
25          }
26
```

Para a questão do CPF, encontramos também alguma dificuldade na questão da apresentação da saída, pois o código só mostrava a forma como a colocávamos, não sendo automático.

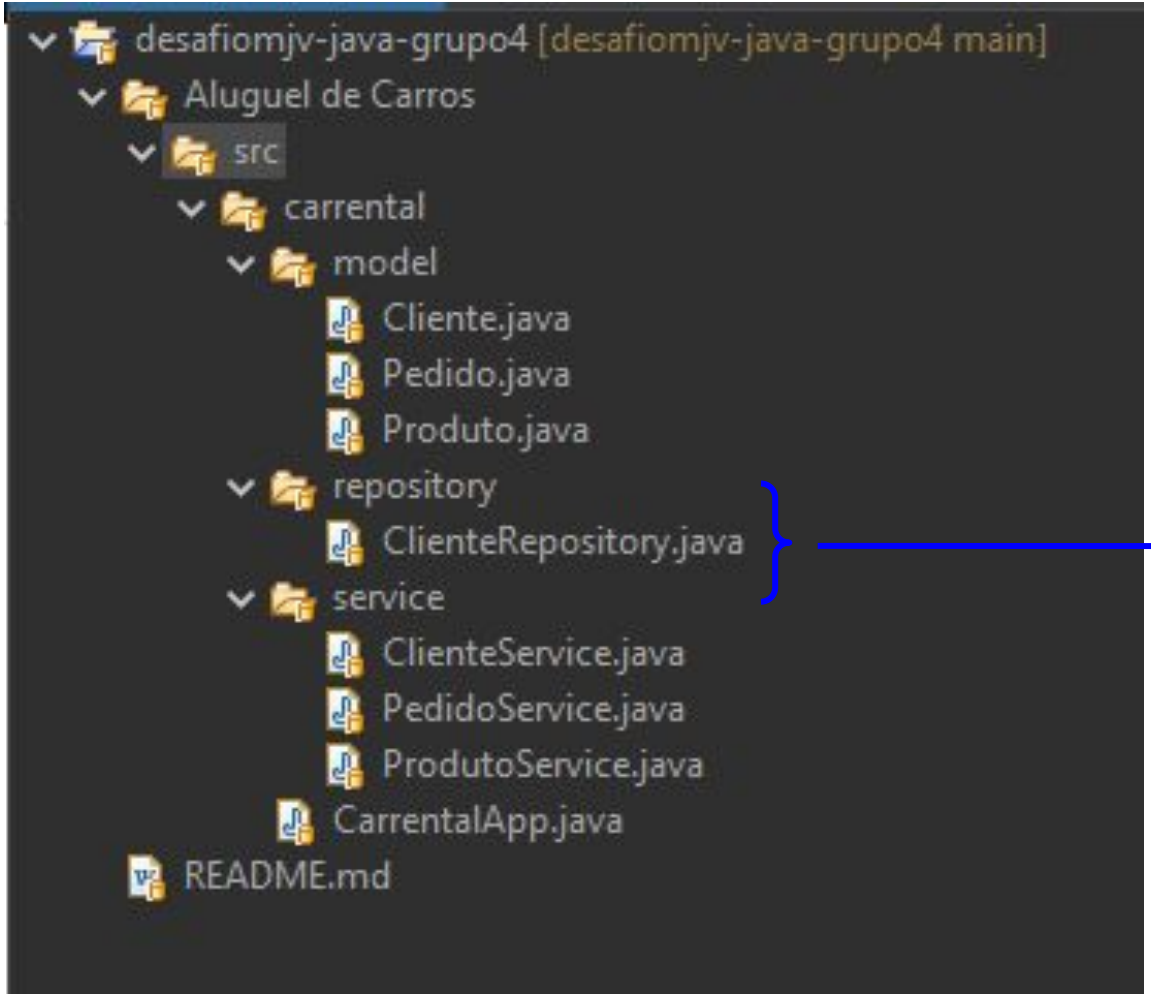
Após pesquisas, chegamos ao código acima, separando por substring e a pontuação desejada, similar ao que pudemos encontrar na documentação oficial.

```java
package carrental.service;

import java.util.Calendar;
import java.util.Date;

public class PedidoService {

    public static Date devolucao(Date dataPedido, int qntDias){
        Calendar cal = Calendar.getInstance();
        cal.setTime(dataPedido);
        cal.add(Calendar.DAY_OF_MONTH, qntDias);
        Date dataDevolucao = cal.getTime();
        return dataDevolucao;
    }
```

Ainda na package de serviço, também utilizamos às classes Calendar e Date com formatação para podermos criar às informações do método “CriarPedido” aliando a data do pedido a quantidade de dias desejada pelo cliente para o aluguel

```java
package carrental.service;

public class ProdutoService {


        public String formatarPlaca(String placa) {

                return(placa.toUpperCase().substring(0, 3) + "-" + placa.substring(3, 7) );
        }

```

Por fim, para a placa do carro, usamos um Formatter mais simples (similar à aplicação no CPF) com substrings, a pontuação desejada e com o .toUpperCase para mitigar erros de digitação de minúsculas, padronizando como as placas reais.



Como último package, criamos o “repository” para incluir o set de clientes e produtos. Eles serviram para os testes, embora não tenham sido requeridos no desafio

```java
package carrental.repository;

import java.util.Set;
import java.util.HashSet;


import carrental.model.Cliente;

public class ClienteRepository {

	private Set <Cliente> clientes = new HashSet<Cliente>();

	public Set<Cliente> getClientes() {
		return clientes;
	}

	public void cadastrarCliente(Cliente cliente) {
		clientes.add(cliente);


	}
```

```java
package carrental.repository;

import java.util.HashSet;
import java.util.Set;

import carrental.model.Produto;

public class ProdutoRepository {
	private Set <Produto> produtos = new HashSet<Produto>();

	public Set<Produto> getProduto() {
		return produtos;
	}

	public void criarProduto(Produto produto) {
		produtos.add(produto);

	}
}
```

Amostra das telas de ClienteRepository e ProdutoRepository

Métodos mais utilizados e seus destaques

Como último ponto do desafio que tinha como foco “apresentar no mínimo 04 métodos mais utilizados destacando o seu contrato (tipo retorno + nome + parâmetros)”, nós verificamos e utilizamos:

.toUpperCase()

Sintaxe: **public String toUpperCase(Locale loc) / public String toUpperCase()**

Método de classe string que converte todos os caracteres em letras maiúsculas.

Parâmetro: valor "locale" a ser aplicado enquanto converte todos os caracteres;

Retorno: retorna letras maiúsculas.

.getInstance

Sintaxe: public static Signature getInstance(String algorithm)

Usado para retornar a assinatura de um objeto que implementa a assinatura específica do algoritmo.

Parâmetro: este método pega o nome padrão do algoritmo como parâmetro;

Retorno: retorna a nova assinatura do objeto (new Signature object).

Métodos mais utilizados e seus destaques

.setTime

Sintaxe: **public void setTime(long time)**

Configura um date object, que aceita um único parâmetro que especifica o número em milissegundos.

Parâmetro: função aceita um único parâmetro de tempo (time) que especifica conforme acima;

Retorno: não tem valor de retorno.

.substring

Sintaxe: **public String substring(int beginIndex, int endIndex)**

Usado para extrair uma porção de string de uma string dada. Cria um novo objeto string sem alterar a original

Parâmetro: início e final do índice (beginIndex, endIndex), para efetuar a busca do inicio e do fim;

Retorno: um novo objeto string resultando da parte da string original cortada da base do índice (de acordo com begin e end)

```java
package carrental;

import java.sql.Date;
import java.time.LocalDate;
import java.util.Set;
import carrental.model.Cliente;
import carrental.model.Pedido;
import carrental.model.Produto;
import carrental.print.MetodoPrint;
import carrental.repository.ClienteRepository;
import carrental.repository.ProdutoRepository;
import carrental.service.ClienteService;
import carrental.service.PedidoService;
import carrental.service.ProdutoService;

public class CarrentalApp {

	public static void main(String[] args) {
		ClienteRepository repository = new ClienteRepository();



		Cliente Joana = new Cliente("Joana", "Joana@gmail.com", "123456", "Rua:ABS, 123", "10/10/1985", "00000000000");
		repository.cadastrarCliente(Joana);

		ProdutoRepository repositoryP = new ProdutoRepository();
		Produto carro = new Produto("Renault Sandeiro", "oqz1917", "Cinza chumbo", 10, 50.00, 2.55);
		repositoryP.criarProduto(carro);
		   Set<Cliente> clientes = repository.getClientes();
		   Set<Produto> produtos = repositoryP.getProduto();



		   ClienteService service = new ClienteService();
```

Ao final ficamos com essa tela na classe principal

```
40                  Joana.setCpf(service.formatarCPF(Joana.getCpf()));
43                  ProdutoService serviceP = new ProdutoService();
44                  carro.setPlaca(serviceP.formatarPlaca(carro.getPlaca()));
45                  carro.setValorTotal(carro.getValorAluguel()*carro.getQtdDias());
50              Date d = Date.valueOf(LocalDate.now());
51
52              Date x =null;
53
54
55              Pedido pedido = new Pedido(1, d , 5, x ,Joana, carro);
56
57              PedidoService servicePedido = new PedidoService();
58              pedido.setDataDevolucao(servicePedido.devolucao(d, carro.getQtdDias()));
59
60
61              System.out.println(pedido);
62
63
64              MetodoPrint teste = new MetodoPrint();
65          teste.abrir();
66          teste.escrever(clientes, produtos, pedido);
67          teste.fechar();
68
69
70          }
71
72
73
74  }
```


LIVE
DEMO

Referências

BAELDUNG. Guide to DateTimeFormatter. Disponível em: https://www.baeldung.com/java-datetimeformatter. Acesso em: 23 dez. 2021.

DEVMEDIA. Formatando números com Numberformat. Disponível em: https://www.devmedia.com.br/formatando-numeros-com-numberformat/7369. Acesso em: 26 dez. 2021.

DICAS DE JAVA. Java 8: Como formatar LocalDate e LocalDateTime. Disponível em: https://dicasdejava.com.br/java-8-como-formatar-localdate-e-localdatetime/. Acesso em: 23 dez. 2021.

JAVA T POINT. Java string format(). Disponível em: https://www.javatpoint.com/java-string-format. Acesso em: 26 dez. 2021.

RECEITAS DE CÓDIGO. NumberFormat ou DecimalFormat, formatar números reais em Java. Disponível em: https://receitasdecodigo.com.br/java/numberformat-ou-decimalformat-formatar-numeros-reais-em-java. Acesso em 21 dez. 2021.

ORACLE. Class DateTimeFormatter. Disponível em: https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html. Acesso em: 21 dez. 2021.

ORACLE. Customizing formats. Disponível em: https://docs.oracle.com/javase/tutorial/i18n/format/decimalFormat.html. Acesso em: 21 dez. 2021.

STACKOVERFLOW. Formatar double em Java. Disponível em: https://pt.stackoverflow.com/questions/55720/formatar-double-em-java. Acesso em: 26 dez. 2021.