ESTGF ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO DE FELGUEIRAS IPP – Instituto Politécnico do Porto	**LEI – Licenciatura em Engenharia Informática** **PP – Paradigmas de Programação** 2º Semestre ■ Docentes: BMO, RJS, TSM e Dcarneiro Ficha Prática 5

Documentação complementar:

- Classes
- Objetos
- Informação adicional

Parte 1

Exercício 1

Desenvolva uma aplicação que permita gerir as despesas pessoais de um utilizador (seguindo o mesmo contexto da ficha prática 4). Cada utilizador é identificado pelo seu código, nome, e-mail, data de nascimento e um conjunto de despesas associadas. As despesas são identificadas por um número, tipo: Automóvel, Alimentar ou Outro (por defeito é assumido o tipo: outro), o valor gasto e a data (dia, mês, ano) da despesa.

Resolução parcial:
Num projecto cujo nome seja "`pp_fp05`", num *package*: `pp_fp05.Expenses`criar uma classe (Java Class...) com o nome `Expense` de forma similar à Figura 1.

```
public class Expense {

    /**
     * Número de identificação da despesa ({@link Expense})
     */
    protected int number;
    /** Tipo da despesa ({@link Expense}) ...3 lines */
    protected String type;
    /** Valor da despesa ({@link Expense}) ...3 lines */
    protected float value;
    /** Data da despesa ((@link Expense}) ...3 lines */
    protected Date data;

    /**
     * Método construtor para a criação de uma instância de @Expense, tendo por
     * base todos os atributos de despesa ({@link Expense})
     *
     * @param tempNumber Número da despesa ({@link Expense})
     * @param tempType Tipo da despesa ({@link Expense})
     * @param tempValue Valor em euros da despesa realizada ({@link Expense})
     * @param tempDate Data({@link Expense}) da despesa realizada
     * ({@link Expense})
     */
    Expense(int tempNumber, String tempType, float tempValue, Date tempDate) {
        number = tempNumber;
        type = tempType;
        value = tempValue;
        data = tempDate;
    }

    Expense(int tempNumber, String tempType, float tempValue, Date tempDate, String currencyType) {...18 lines }
```

Figura 1. Resolução parcial da classe Expense

1.1. Considerando o excerto apresentado na Figura 1, complemente a estrutura da classe Expense. Garanta a integridade dos dados que podem ser armazenados de acordo com a classe apresentada;

```java
public class User {
    private static final int ID_SIZE = 3;
    /**
     * Identificação do utilizador, constituído por três caracteres
     */
    protected char[] id;
    /** Nome do utilizador ({@link User}) ...3 lines */
    protected String name;
    /** Email do utilizador ({@link User}) ...3 lines */
    protected String email;
    /** Data de nascimento (dia/mes/ano) do utilizador ({@link User}) ...3 lines */
    protected Date birthDate;
    /** Despesas ({@link Expense}) do utilizador ({@link User}) ...3 lines */
    protected Expense[] expenses;

    /**
     * Método construtor para a criação de uma instância de {@link Expense},
     * tendo por base todos os atributos de {@link User}
     *
     * @param tempId Identificação do utilizador contituída por 3 carateres
     *
     * @param tempName Nome do utilizador ({@link User})
     * @param tempEmail Email do utilizador ({@link User})
     * @param tempBithDate Data de nascimento ({@link Expense}) do utilizador
     * ({@link User})
     * @param tempExpenses Coleção de despesas ({@link Expense}) do utilizador
     * ({@link User})
     */
    public User(char[] tempId, String tempName, String tempEmail, Date tempBithDate, Expense[] tempExpenses)
```

Figura 2. Resolução parcial da classe User

1.2. Complemente o excerto apresentado na **Figura 2**;

1.3. No mesmo *package*, crie uma classe (*Java Main Class...*) com o nome `ExpensesDemo` (`ExpensesDemo.java`) que implemente o método `main()`. Teste as classes implementadas com pelo menos dois utilizadores e duas despesas cada;

1.4. Implemente agora a nova classe (*Java Main Class...*) `ExpensesDemo` num *package* com o nome `pp_fp05.Demo`. Qual é razão de surgirem erros de compilação? Como poderão ser resolvidos?

1.5. Altere a classe `Expense` de forma a receber a moeda em que as despesas foram realizadas. De seguida, e utilizando a(s) classe(s) do *package* `exchange` implementada na ficha prática anterior, garanta que o valor armazenado se encontra em Euros;

1.6. Altere a classe `User` de forma a diferenciar os utilizadores gratuitos (*Free*) e os que contribuíram monetariamente para a aplicação (*Premium*), armazenando o valor das suas contribuições numa determinada data. O utilizador apenas poderá usufruir do estatuto premium se o valor das suas contribuições for de pelo menos 1 euro. A contribuição de pelo menos um 1 euro permite ao utilizador usufruir deste estatuto durante 5 dias. Mediante o valor da contribuição, o sistema deverá ser capaz de determinar o número de dias com este estatuto.

1.7. Realize as alterações necessárias para que os utilizadores *Premium* possam obter uma listagem das despesas de acordo com um determinado intervalo de tempo e o tipo das despesas que pretende visualizar. Deverá também ser possível visualizar dados gerais como:

- O tipo de despesa mais dispendiosa num determinado intervalo de tempo;
- A despesa mais elevada de cada mês de um determinado ano.
- O valor total gasto (no total das despesas) por mês num determinado ano.

1.8. Realize as alterações necessárias para que o utilizador *Premium* possa aceder a qualquer momento ao total e ao número de despesas de todos os utilizadores.

Documente o código devidamente e gere o JavaDoc para o projeto utilizado

Exercício 2

Desenvolver uma aplicação que permita armazenar informação sobre um CD com 15 músicas, sendo que cada música é caracterizada por: número da faixa, nome, duração (em segundos) e autor.

2.1. Assinar e implementar o método construtor que permita alterar o estado dos quatro atributos de `Track`.

2.2. Criar na package `pp_fp05.cd` uma classe (Java Class...) com o nome `CD` (`CD.java`) .

2.3. Declarar os atributos de `CD`: `name, year, editor` e `tracks` (conjunto de `Track's`).

2.4. Criar na package `pp_fp05.cd` uma classe (Java Main Class...) que implemente o método `main()` com o nome `CDDemo` (`CDDemo.java`).

2.5. Implementar o método `main()` da classe `CDDemo` declarando a variável necessária (`cd`), inicializando a primeira e a última posição do atributo `tracks` (o qual é membro de `CD`) e produzindo alguma saída de informação. Tome por base o excerto da Figura 3.

```
public class CDDemo {

    public static void main(String[] args) {
        Track[] t = {
            new Track(1, "Ho Hey", 90, "Lumineers"),
            new Track(2, "Stubborn Love", 105, "Wesley Schultz")
        };
        CD cd = new CD("The Lumineers", 2012, "Dualtone Records", 15, t);

        System.out.println("Nome do CD: " + cd.name);
        System.out.println("Ano de lançamento: " + cd.year);
        System.out.println("Editora: " + cd.editor);

        for (Track track : t) {
            System.out.println("Título da música número "
                    + track.number + " do CD: " + track.title);
            System.out.println("Duração: " + track.duration);
            System.out.println("Autor da musica: " + track.author);
        }

    }
}
```

Figura 3: Classe **CDDemo**.

Executar a classe `CDDemo`. O resultado deve ser similar ao indicado na Figura 4.

```
Nome do CD: The Lumineers
Ano de lançamento: 2012
Editora: Dualtone Records
Título da música número 1 do CD: Ho Hey
Título da música: Ho Hey
Duração da música: 90
Autor da música: Lumineers
Título da música número 15 do CD: Stubborn Love
Título da música: Stubborn Love
Duração da música: 105
Autor da música: Wesley Schultz
```

Figura 4: Resultado da execução da classe **CDDemo**.

2.6. Como fazer para que cada CD criado possua um dado número de faixas, ou seja, não contenha sempre 15? Implemente as alterações necessárias e teste.

Parte 2

Exercício 1

3.1. No package `pp_fp05.cd`, implemente uma classe `Author` que represente um autor(es) de uma música. Cada autor é representado pelos atributos: name, age, address, NIF e NIB para onde irão reverter os lucros da venda de músicas. De realçar que os autores podem-se registar como vendedores, devendo disponibilizar todos os dados referidos anteriormente, ou como não vendedores disponibilizando a sua música gratuitamente. Para isso apenas precisam de fornecer nome e idade.
Note que, sempre que seja criada uma instância de `Author` os atributos da mesma terão de ser inicializados através do construtor. Um CD poderá ter vários autores que podem ser de dois tipos distintos.

3.2. Altere a classe `Track` de modo a utilizar esta nova classe para um máximo de 5 autores.

3.3. Altere o método `main` na classe `CDDemo` (no package `fp_fp05.cd`) que permita listar os dados dos autores para cada música.

Exercício 2

4.1. Adicione na classe `CD` o atributo `Price` e disponibilize um método construtor apropriado. Lembre-se que um CD poderá não possuir um preço associado.

4.2 Adicione no package `fp_fp05.store`, uma classe `User`, que permita armazenar dados relativos a um utilizador da loja de música, com os atributos: name, age e email. Note que, sempre que seja criada uma instância de `User` os atributos da mesma terão de ser inicializados através do construtor.

4.3. No package `fp_fp05.store` defina uma nova classe `Sale` que permita disponibilizar informação sobre a venda: saleId, day/month/year, lista de CD's associados à compra respectiva, assim como o preço final (TotalPrice) a pagar (o preço é dependente do preço dos CD's contidas na compra).

4.4. Com base na classe `CDDemo`, crie a classe `StoreDemo` de forma a testar e a imprimir a informação que se pode extrair de cada venda realizada: data da venda, lista CD's comprados e o preço final a pagar.

Gere o JavaDoc para o projecto utilizado na resolução desta ficha de trabalho.