

Sumário

- Classes and Objects: <http://java.sun.com/docs/books/tutorial/java/javaOO/index.html>
- Objectos
- Métodos Construtores
- Variáveis de Instância
- Strings

Documentação complementar:

- [Classes](#)
- [Objetos](#)
- [Informação adicional](#)

Exercício 1

Recorde a aplicação desenvolvida na ficha prática 4 (e ficha prática 4b) cujo objetivo era armazenar as despesas realizadas durante um mês (ou ano). Pretende-se o desenvolvimento de uma aplicação semelhante que permita gerir as despesas de um ou mais utilizadores.

Considere que um utilizador possui a seguinte informação:

- Código (único para cada utilizador e incremental)
- Nome
- E-mail
- Data de nascimento (em formato *String* – “YYYY-MM-DD”)
- Conjunto de despesas associadas.

A despesa contém os seguintes valores:

- Número de identificação (que deverá ser único para cada despesa, independentemente do utilizador a que esteja associada)
- Tipo (automóvel, alimentar ou outro – por omissão é considerado outro)
- Valor gasto
- Data da despesa (em formato *String* – “YYYY-MM-DD”)

Resolução parcial:

1. Crie um novo projeto cujo nome seja “pp_fp05”
2. Adicione um *package* “pp_fp05.expenses”
3. Crie uma nova classe designada *Expense* de forma idêntica à Figura 1

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;

1.2) Complemente o excerto da classe *User* apresentado na Figura 2;

1.3) No mesmo *package* crie uma *main* classe (Java Main Class) com o nome *ExpensesDemo*. Teste as classes implementadas nos pontos anteriores com pelo menos dois utilizadores e duas despesas cada;

1.4) Mova a *main* classe para um novo *package* com o nome *pp_fp05.Demo*. Por que razão surgiram erros de compilação? Como poderão ser resolvidos?

```
package pp_ficha05.ex01;

/** <h1> Classe utilizada para armazenar a informação relativa a uma despesa de ...16 lines */
public class Expense {

    /** Número de identificação da {@link Expense} ...3 lines */
    protected int id;

    /** Tipo da {@link Expense} ...3 lines */
    protected String type;

    /** Valor da {@link Expense} ...3 lines */
    protected float amount;

    /** Data de realização da {@link Expense} ...3 lines */
    protected String date;

    /** Método construtor para a criação de uma instância de {@link Expense} ...7 lines */
    public Expense(String tempType, float tempAmount, String tempDate) {
        id = nextId++;
        type = tempType;
        date = tempDate;
        amount = tempAmount;
    }
}
```

Figura 1 - Resolução parcial da classe Expense

```
package pp_ficha05.ex01;

/** <h1> Classe utilizada para armazenar a informação relativa a um utilizador ...16 lines */
public class User {

    /** Tamanho, por omissão, definido para a criação do array de {@link Expense} ...3 lines */
    protected final int EXPENSES_SIZE = 31;

    /** Número de identificação do {@link User} ...3 lines */
    protected int id;

    /** Nome do {@link User} ...3 lines */
    protected String nome;

    /** Email do {@link User} ...3 lines */
    protected String email;

    /** Data de nascimento do {@link User} ...4 lines */
    protected String birthDate;

    /** Array com o conjunto de {@link Expense} do {@link User} ...3 lines */
    protected Expense[] expenses;

    /**
     * Construtor do {@link User} que inicializa o array de {@link Expense} com
     * o tamanho por omissão {@value #EXPENSES_SIZE}
     *
     * @param tempNome nome do {@link User}
     * @param tempEmail email do {@link User}
     * @param tempBirthDate data de nascimento do {@link User}
     */
    public User(String tempNome, String tempEmail, String tempBirthDate) {
        expenses = new Expense[EXPENSES_SIZE];
        id = nextId++;
        nome = tempNome;
        email = tempEmail;
        birthDate = tempBirthDate;
    }
}
```

Figura 2 - Resolução parcial da classe User

Exercício 2

Desenvolva uma aplicação que permita armazenar informação sobre um CD com 15 músicas. Cada CD terá a seguinte informação:

- Nome da banda
- Nome do CD
- Tempo total (em segundos)
- Ano de lançamento
- Editora
- Conjunto de elementos que constituem a banda
- Conjunto de músicas

Para complementar a informação anteriormente descrita, cada música é composta por

- Número da faixa
- Nome da faixa
- Duração em segundos
- Nome do autor

Cada elemento da banda terá a seguinte informação:

- Nome do artista
- Nacionalidade
- Data de nascimento (utilize uma *string* com o formato YYYY-MM-DD)


- 2.1) Implemente a classe `Track` com um método construtor que permita alterar o estado dos seus quatro atributos.
- 2.2) Crie e implemente a classe `CD` num novo *package* `pp_fp05.cd`
- 2.3) Crie e implemente a classe `Artist`
- 2.4) No *package* `pp_fp05.cd` crie uma *main* classe com o nome `CDDemo`. Nesta declare uma variável `cd` e inicialize a primeira e última posição do atributo `tracks`. Tenha em atenção o exemplo apresentado na Figura 3 cujo resultado de execução é apresentado na Figura 4
- 2.5) Altere o programa desenvolvido para que cada CD não fique limitado a 15 faixas.

```
public class EX02 {  
    /**...3 lines */  
    public static void main(String[] args) {  
        Artist a1 = new Artist("Artista 1", "1977-03-04", "German");  
        Artist[] artists = {a1};  
  
        Track t1 = new Track(1, "Ho Hey", 90, "Lumineers");  
        Track t2 = new Track(2, "Stubborn Love", 105, "Wesley Schlitz");  
  
        CD cd = new CD("The Lumineers", "The Lumineers", 2012, "Dualtone Records", 195, artists);  
        cd.tracks[0] = t1;  
        cd.tracks[14] = t2;  
  
        System.out.println("Nome do cd: " + cd.cdName);  
        System.out.println("Ano de lançamento: " + cd.year);  
        System.out.println("Editora: " + cd.editor);  
  
        int nTracks = cd.tracks.length;  
        for (int i = 0; i < nTracks; i++) {  
            if (cd.tracks[i] != null) {  
                Track t = cd.tracks[i];  
                System.out.println("Música número: " + t.number + " com título: " + t.name);  
                System.out.println("Duração (em segundos): " + t.duration);  
                System.out.println("Autor da música: " + t.authorName);  
            }  
        }  
    }  
}
```

Figura 3 - Exemplo da classe com método main

```
Nome do cd: The Lumineers  
Ano de lançamento: 2012  
Editora: Dualtone Records  
Música número: 1 com título: Ho Hey  
Duração (em segundos): 90  
Autor da música: Lumineers  
Música número: 2 com título: Stubborn Love  
Duração (em segundos): 105  
Autor da música: Wesley Schlitz
```

Figura 4 - Exemplo de execução da classe EX02

 <div> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	<div> LEI/LSIRC PP - Paradigmas de Programação 2º Semestre ■ Docentes: RJS, BMO, CDF, MFG, OAO Ficha Prática 5 – 2023/2024 </div>
--	--

Exercício 3

3.1. No *package* `pp_fp05.cd`, implemente uma classe `Author` que represente um autor de uma música. Cada autor é representado pelos atributos:

- nome
- idade
- morada
- NIF
- NIB (para onde irão reverter os lucros da venda de músicas)

Um autor pode registar-se como vendedor ou gratuito. Caso seja considerado vendedor terá de disponibilizar todos os dados referidos anteriormente. Se for registado como gratuito, então apenas será necessário nome e idade. 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 4

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:

- nome
- idade
- email.

4.3. No *package* `fp_fp05.store` defina uma nova classe `Sale` que permita disponibilizar informação sobre a venda:

- id da venda (que deverá ser único para cada compra, independentemente do utilizador)
- data da compra (string no formato “YYYY-MM-DD”)
- lista de `CD`’s associados à compra
- preço final 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(s) projeto(s) utilizado(s) na resolução desta ficha de trabalho.