Prática 5 Classes

Tópicos

- Classes, instâncias e metodologias orientadas a objetos
- Construtores, atributos e métodos
- Métodos especiais (`toString(), equals(), get…(), set…(), ...`)

Exercício 5.1

Crie uma classe que permita modelar uma data (class Date).

Esta classe deve conter os seguintes métodos estáticos:
- um método booleano que indique se o valor inteiro que represente um mês ([1;12]) é válido: validMonth(int month)
- um método inteiro que devolva o número de dias de um determinado mês, num determinado ano: monthDays(int month, int year)
- um método booleano que indique se um ano é bissexto: leapYear(int year)
- um método booleano que indique se uma data composta por dia, mês e ano, é válida: valid(int day, int month, int year)

(Nota: No desenvolvimento destes métodos aproveite os métodos desenvolvidos nas aulas anteriores.)

A classe deve também permitir instanciar objetos que representem uma data específica (válida). Nesse sentido, considere que a representação interna do objeto é composta por três atributos inteiros (day, month, year).
Deve ser possível aplicar externamente as seguintes operações sobre objetos deste tipo:
- definir uma data: set(int day, int month, int year);
- consultar os valores do dia, mês e ano (day, month, year);
- incrementar a data (increment);
- decrementar a data (decrement);
- método toString que devolva a data no formato AAAA-MM-DD.

A classe deve ter um construtor que defina uma data (válida) indicando um dia, mês e ano.

(Nota: desenvolva a classe garantindo que não é possível que nenhum dos seus objetos represente uma data inválida [por exemplo, 31 de fevereiro de 2022].)

Para testar esta classe, crie um programa de teste, com o seguinte menu:

```
Date operations:
1 - create new date
2 - show current date
3 - increment date
4 - decrement date
0 - exit
```

Exercício 5.2

Construa uma classe que represente um calendário.
Esta classe deve fazer uso da classe Date desenvolvida no exercício anterior, e deve incluir:
- um construtor que recebe o ano e o dia da semana (entre 1-domingo e 7-sábado) em que começa;
- métodos que devolvem esses dados (consultas/getters): year() e firstWeekdayOfYear();
- um método que devolva o dia da semana em que começa um dado mês (no ano do calendário): firstWeekdayOfMonth(month);
- um método que imprima um mês de calendário: printMonth(month);
- método toString que devolva o calendário para todo o ano:

```
    January 2022
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31

    February 2022
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28

...
```

Para testar esta classe, crie um programa de teste, com o seguinte menu:

```
Calendar operations:
1 - create new calendar
2 - print calendar month
3 - print calendar
0 – exit
```

Exercício 5.3

Implemente classes que permitam modelar as seguintes formas geométricas:

- Círculo, caracterizado pelo seu *raio*;
- Triângulo, caracterizado pela dimensão dos seus lados (*lado1, lado2, lado3)*;
- Retângulo, caracterizado por *comprimento* e *altura.*

Garanta ainda as seguintes especificações:

a) crie classes que representem cada uma das figuras geométricas, implementando construtores e métodos adequados para cada classe.
b) adicione todos os métodos especiais importantes (`toString(), equals(), get…(), set…(), ...`);
c) nos construtores e métodos modificadores (set...), verifique pré-condições adequadas:

raio e lados têm de ser valores positivos e lados do triângulo têm de satisfazer a desigualdade triangular;

d) implemente um método para calcular a área de cada tipo de figura (para triângulo, ver fórmula de Heron);

e) implemente um método para calcular o perímetro de cada tipo de figura;

f) implemente um programa que teste todas as classes e métodos criados; o programa deve criar um conjunto de figuras, especificadas pelo utilizador através de um menu, listá-las (o método `toString()` deve mostrar o tipo e características da figura), e comparar os pares de figuras do mesmo tipo (método `equals()`).

Exercício 5.4

Pretende-se construir um sistema de informação simplificado para a gestão da biblioteca de uma universidade. A biblioteca contém um catálogo de livros e um conjunto de utilizadores (só alunos). Todos os utilizadores são identificados pelo seu número mecanográfico, nome e curso. Os livros são caraterizados por um ID (numérico e sequencial, começando em 100), título e tipo de empréstimo (CONDICIONAL ou NORMAL). Comece com as definições seguintes:

```
public class Utilizador {
   private String nome;
   private int nMec;
   private String curso;
    // .....
}
```

```
public class Livro {
    private int id;
    private String titulo;
    private String tipoEmprestimo;
    // .....
}
```

Faça uso de modificadores de acesso para garantir que todos os atributos das classes não estão acessíveis do exterior. Em caso de necessidade, defina novos atributos para responder aos requisitos do enunciado. Teste as classes desenvolvidas com o programa seguinte:

```
import java.util.ArrayList;

public class Ex52 {

   public static void main(String[] args) {

      // Para o conjunto de Livros vamos criar um vetor de 10 posições
      // Este vetor tem uma dimensão fixa pelo que se for necessário guardar
      // mais livros, teremos de criar um vetor de maior dimensão.
      Livro catalogo[] = new Livro[10];
      catalogo[0] = new Livro("Java 8", "CONDICIONAL");
      catalogo[1] = new Livro("POO em Java 8");
      catalogo[2] = new Livro("Java para totós", "NORMAL");
      System.out.println("ID = " + catalogo[1].getId() + ", "
                         + catalogo[1].getTitulo());
      catalogo[2].setTipoEmprestimo("CONDICIONAL");

      for (int i = 0; i < catalogo.length; i++) { // usando o indice do vector
         if (catalogo[i] != null) // porque o vector catalogo não está cheio
            System.out.println(catalogo[i]);
      }

      // Para o conjunto de utilizadores usamos a classe java.util.ArrayList
      // É uma implementação de um vetor com tamanho variável
      ArrayList<Utilizador> alunos = new ArrayList<>();
```

```
        alunos.add(new Utilizador("Catarina Marques", 80232, "MIEGI"));
        alunos.add(new Utilizador("Joao Silva", 90123, "LEI"));
        alunos.get(1).setnMec(80123);

        for (Utilizador u : alunos) { // usando foreach
              System.out.println(u);
        }
    }
}
```

Cujo resultado da sua execução deve ser:

```
ID = 101, POO em Java 8
Livro 100; Java 8; CONDICIONAL
Livro 101; POO em Java 8; NORMAL
Livro 102; Java para totós; CONDICIONAL
Aluno: 80232; Catarina Marques; MIEGI
Aluno: 80123; Joao Silva; LEI
```

Exercício 5.5

Utilizando as classes desenvolvidas no exercício anterior, implemente um programa que permita gerir os utilizadores e empréstimos numa biblioteca. Comece por construir, de uma forma interativa, o seguinte menu:

```
1 - inscrever utilizador
2 - remover utilizador
3 - imprimir lista de utilizadores
4 - registar um novo livro
5 - imprimir lista de livros
6 - emprestar
7 - devolver
8 - sair
```

Condições adicionais:

a) Recomenda-se que as operações de empréstimo e devolução sejam efetuadas com base no ID do livro e no número mecanográfico do aluno.

b) Cada aluno só poderá requisitar simultaneamente um máximo de 3 livros. Deve modificar a classe utilizador para poder guardar os IDs dos livros requisitados, bem como a classe livro para indicar a sua disponibilidade.

c) Para simplificar, considere que apenas existe uma cópia de cada livro e que os livros com tipo de empréstimo CONDICIONAL não podem ser requisitados.

d) Para guardar o catálogo de livros e a lista de alunos, utilize vetores, considerando que no máximo a biblioteca pode ter 100 livros e 100 utilizadores.