

Aula prática nº 5

Tópicos

- Herança simples
- Referências **super** e **this**

Exercícios

1. Construa a classe `Pessoa` que é caracterizada pelo nome, número do cartão do cidadão e data de nascimento. Comece com as definições seguintes:

```
public class Pessoa {
    String nome;
    int cc;
    Data dataNasc;
    // .....
}

public class Data{
    int dia;
    int mes;
    int ano;
    // .....
}
```

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 métodos a incluir na interface pública da classe.

Crie métodos adequados para permitir a inicialização dos seus atributos no momento de criação de cada objeto:

```
Data d = new Data(5, 10, 1988);
Pessoa p = new Pessoa("Ana Santos", 98012244, d);
```

Construa uma nova classe `Aluno`, derivada da classe `Pessoa`, acrescentando os métodos e atributos necessários para aceder e guardar o número mecanográfico (`int`) e a data de inscrição (`Data`) na instituição de ensino. Note que o número mecanográfico deverá ser atribuído automaticamente (e sequencialmente a partir do 100) quando da criação de um novo aluno.

A estrutura simplificada das classes deverá ser a seguinte:

```
public class Pessoa {
    //...
    String getName(){...}          // retorna o nome da pessoa
}

public class Aluno extends Pessoa{
    //... definição de atributos

    Aluno(String iNome, int iBI, Data iDataNasc, Data iDataInsc);
    Aluno(String iNome, int iBI, Data iDataNasc);
                                // nota: neste caso deve assumir a data atual
    int getNMec() {...}          // retorna o número mecanográfico
    // ... acrescentar métodos necessários
}
```

Crie a classe *Bolseiro*, derivada da classe *Aluno*, que deverá incluir um atributo com o montante da bolsa. Defina novos métodos ou reescreva os métodos que julgar conveniente. Acrescente métodos *get/set* associados ao valor da bolsa.

Implemente o método "`@Override public String toString()`" em todas as classes. Por exemplo, para a classe *Pessoa*, deve retornar:

"Ana Santos, CC: 98012244 Data: 5/10/1988"

Teste o trabalho desenvolvido com o seguinte programa:

```
public class Test {
    public static void main(String[] args) {
        Aluno al = new Aluno ("Andreia Melo", 9855678,
            new Data(18, 7, 1990), new Data (1, 9, 2014));
        Bolseiro bls = new Bolseiro ("Igor Santos", 8976543, new Data(11, 5, 1985));
        bls.setBolsa(745);

        System.out.println("Aluno:" + al.getName());
        System.out.println(al);

        System.out.println("Bolseiro:" + bls.getName() + ", NMec: " + bls.getNMec()
            + ", Bolsa:" + bls.getBolsa());

        System.out.println(bls);
    }
}
```

- Utilizando herança, reescreva o programa desenvolvido na aula 4 relativamente às figuras geométricas. Pretende-se trabalhar com as seguintes classes: *Ponto*, *Figura*, *Circulo*, *Quadrado* e *Retângulo*. Redefina o método `equals()` para que seja possível comparar se duas figuras são do mesmo tipo. Atenção que não é necessário neste caso verificar se os atributos dos objetos são iguais.

Exercício adicional – explorar uso de arrays

- Construa uma classe *Conjunto* que guarda um conjunto de números inteiros (estes não podem repetir). Utilize arrays e implemente as funções seguintes:
 - `void insert(int n);` – para inserir um elemento novo no conjunto. Caso este elemento já exista, a função não faz nada. Inicialmente não se sabe quantos elementos vamos inserir.
 - `boolean contains(int n);` – para indicar se um dado elemento está no conjunto.
 - `void remove(int n);` – para remover um elemento do conjunto. Caso este elemento não se encontre no conjunto, a função não faz nada.
 - `void empty();` – para apagar todos os elementos do conjunto.

- `String toString()` ; - para converter os elementos do conjunto numa `String`.
- `int size()` ; - para calcular o número de elementos no conjunto.
- `Conjunto unir(Conjunto add)` ; - para construir um conjunto novo que representa a união de dois conjuntos. O conjunto resultante não deve conter elementos repetidos.
- `Conjunto subtrair(Conjunto dif)` ; - para construir um conjunto novo que representa a diferença do `this` e dos elementos do conjunto representado pelo objeto `dif`.
- `Conjunto interset(Conjunto inter)`; - para construir um conjunto novo que representa a intersecção do `this` com os elementos do conjunto representado pelo objeto `inter`. O conjunto resultante não deve conter elementos repetidos.

Teste a classe desenvolvida com a função *main* seguinte:

```
public static void main(String[] args) {
    Conjunto c1 = new Conjunto();
    c1.insert(4); c1.insert(7); c1.insert(6); c1.insert(5);

    Conjunto c2 = new Conjunto();
    int[] test = { 7, 3, 2, 5, 4, 6, 7};
    for (int el : test) c2.insert(el);
    c2.remove(3); c2.remove(5); c2.remove(6);

    System.out.println(c1);
    System.out.println(c2);

    System.out.println("Número de elementos em c1: " + c1.size());
    System.out.println("Número de elementos em c2: " + c2.size());

    System.out.println("c1 contém 6?: " + ((c1.contains(6) ? "sim" : "não")));
    System.out.println("c2 contém 6?: " + ((c2.contains(6) ? "sim" : "não")));

    System.out.println("União:" + c1.unir(c2));
    System.out.println("Intersecção:" + c1.interset(c2));
    System.out.println("Diferença:" + c1.subtrair(c2));

    c1.empty();
    System.out.println("c1:" + c1);
}
```

Os resultados devem ser os seguintes (a ordem dos elementos é irrelevante):

```
4 7 6 5
7 2 4
Número de elementos em c1: 4
Número de elementos em c2: 3
c1 contém 6?: sim
c2 contém 6?: não
União:4 7 6 5 2
Intersecção:7 4
Diferença:6 5
c1:
```