Prática 6 Classes e herança

Objetivos

- Compreender as relações entre classe
- Compreender e utilizar as referências super e this

Tópicos

- Herança simples
- Composição 

Exercício 6.1

Construa a classe *Pessoa* que é caraterizada pelo nome, número do cartão do cidadão e data de nascimento. Comece com as definições seguintes e defina novos métodos a incluir na interface pública da classe. Reutilize a classe *DateYMD* criada na aula anterior.

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

Crie métodos adequados para permitir a inicialização dos seus atributos no momento de criação de cada objeto, garantindo que não é possível criar objetos com estado (valores dos atributos) inválido:

```
DateYMD d = new DateYMD(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 (DateYMD) 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 getNome(){...}         // retorna o nome da pessoa
}

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

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

Crie uma classe *Professor*, derivada de *Pessoa*, que deverá incluir um atributo com indicação da categoria ("Auxiliar", "Associado", "Catedrático") e do departamento.
Crie a classe *Bolseiro*, derivada da classe *Aluno*, que deverá incluir um atributo com a indicação do professor orientador e um atributo com o montante mensal da bolsa. Defina novos métodos ou reescreva os métodos que julgar conveniente. Acrescente métodos *get/set* associados ao orientador e 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 de Nascimento: 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 DateYMD(18, 7, 1990), new DateYMD(1, 9, 2018));
    Professor p1 = new Professor("Jorge Almeida", 3467225, new DateYMD(13, 3, 1967),
"Associado", "Informática");
    Bolseiro bls = new Bolseiro ("Igor Santos", 8976543, new DateYMD(11, 5, 1985), p1,
900);
    bls.setBolsa(1050);

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

    System.out.println("Bolseiro: " + bls.getNome() + ", NMec: "
       + bls.getNMec() + ", Bolsa: " +  bls.getBolsa() + ", Orientador: " +
bls.getOrientador());
    System.out.println(bls);
    }
 }
```

Exercício 6.2 Contactos

Pretende-se criar um sistema que permita gerir uma lista de contactos dos alunos e professores de uma instituição. Cada contacto vai conter um ID (número sequencial começando em 1), a pessoa correspondente, o número de telemóvel e o endereço de email.
Reutilize as classes do exercício anterior e considere que:

- Cada contacto tem de estar associado a uma pessoa;
- Cada contacto deve conter pelo menos o número de telemóvel ou o endereço de email;
- O número de telemóvel deve ser validado, considerando 9 algarismos começados por '9';
- O endereço de email deve ser validado, devendo conter um caracter '@' e terminar com o domínio ('.pt', '.com', ...).

Crie o seguinte menu para gestão dos contactos:

1. Inserir contacto
2. Alterar contacto
3. Apagar contacto

4. Procurar contacto
5. Listar contactos
0. Sair

Considere as seguintes condições de funcionamento:

- Ao inserir um contacto se a pessoa já existir o programa deve perguntar se quer continuar a inserir como novo contacto;
- Alterar/Apagar/Procurar contacto deve permitir pesquisar por número ou nome, caso haja dois iguais, deve mostrar no ecrã e pedir o ID de qual pretende alterar;

Exercício 6.3 Conjuntos

Construa uma classe *Conjunto* que guarda um conjunto de números inteiros (que não se podem repetir). 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 combine(Conjunto add);` - para construir um conjunto novo que representa a união de dois conjuntos. O conjunto resultante não deve conter elementos repetidos.
- `Conjunto subtract(Conjunto dif);` - para construir um conjunto novo que representa a diferença do this e dos elementos do conjunto representado pelo objeto dif.
- `Conjunto intersect(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 pode 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("Interseção:" + 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
Interseção:7 4
Diferença:6 5
c1:
```