~~Aula Prática 9~~

Objectivos

Composição versus Herança.
Utilização alguns padrões de software: Decorador, Singleton e Iterador.

~~Problema 9.1~~

Considere as seguintes entidades:

- Jogador (joga)
- Futebolista (joga, passa, remata, ..)
- Tenista(joga, serve,...)
- Basquetebolista (joga, passa, lança, ..)
- Xadrezista (joga, move, ...)

Desenvolva classes que representem adequadamente o problema. Note que existem Jogadores que praticam mais do que uma modalidade. Utilize classes decoradoras para adicionar dinamicamente comportamentos a cada Jogador.

Teste a solução final com o seguinte código:

```
public static void main(String args[]) {
   JogadorInterface j1 = new Jogador("Rui");
   Futebolista f1 = new Futebolista(new Jogador("Luis"));

   Basquetebolista b1 = new Basquetebolista(
                           new Jogador("Ana"));
   Basquetebolista b2 = new Basquetebolista(j1);
   Basquetebolista b3 = new Basquetebolista(f1);

   Tenista t1 = new Tenista(j1);
   Tenista t2 = new Tenista(new Basquetebolista(
                             new Futebolista(
                                 new Jogador("Bruna"))));

   Xadrezista xd1 = new Xadrezista (new Futebolista(
                                      new Jogador("Paredes")));

   JogadorInterface lista[] = {j1, f1, b1, b2, b3, t1, t2, x1};
   for (JogadorInterface ji: lista)
      ji.joga();
}
```

~~Problema 9.2~~

Com base nas classes Pessoa, ~~Data~~ e ListaPessoas desenvolvidas durante as primeiras aulas construa:

~~a)~~ a classe VectorPessoas que gere uma lista de Pessoas com base num vector que cresce dinamicamente. Inclua os métodos:

```
boolean addPessoa(Pessoa p)
boolean removePessoa(Pessoa p)
int totalPessoas()
```

~~b)~~ Acrescente à classe VectorPessoas a classe interna VectorIterator que implementa a interface Iterador :

```
public interface Iterator {
      boolean hasNext();
      Object next();
      void remove();
}
```

~~c)~~ Crie o método Iterator iterador() que cria um iterador a apontar para a primeira posição do vector interno.

~~d)~~ Repita estes procedimentos para uma nova classe ListaPessoas onde o armazenamento de Pessoas é feito à custa de uma lista ligada.

~~e)~~ Teste as classes desenvolvidas com um código do tipo:

```
public abstract class TesteIterador {

   public static void main(String[] args) {
      VectorPessoas vp = new VectorPessoas();
      for (int i=0; i<10; i++)
         vp.addPessoa(new Pessoa("Bebé no Vector "+i,
                      1000+i, Data.today()));
      Iterator vec = vp.iterator();
      while ( vec.hasNext() )
         System.out.println( vec.next() );

      ListaPessoas lp = new ListaPessoas();
      for (int i=0; i<10; i++)
         lp.addPessoa(new Pessoa("Bebé na Lista "+i,
                      2000+i, Data.today()));
      Iterator lista = lp.iterator();
      while ( lista.hasNext() )
         System.out.println( lista.next() );
   }
}
```