# Ficha Prática 4

António Nestor Ribeiro, Paulo Azevedo, Mário Martins
{anr,af,fmm}@di.uminho.pt
PPIV (LESI)

2005/06

## Objectivos

1. Classes e instâncias. Métodos e variáveis de instância.
2. Criação de classes por composição de outras classes.
3. Classe StringBuffer.
4. Exercícios.

## Exercícios

1. Analise a documentação da classe `StringBuffer`, que permite a implementação de uma String de forma eficiente.

   Permite o mesmo tipo de operações, mas sem criar novas strings para cada alteração. É utilizada internamente pelo compilador ao executar operações de concatenação de Strings do tipo:

   ```
   x = "Viva o " + language + " " + version

   /* tudo na mesma linha. A evitar ;) */
   x = new StringBuffer().append("Viva o ").append(language).append(" ")
       .append(version).toString();
   ```

   Aguns métodos úteis (dos muitos existentes):

   - `append(...)`

- `toString`
- `int indexOf(String a);` // devolve o índice do início da String a na instância.
- `int indexOf(String a, int fromIndex);` // faz o mesmo que o anterior, mas a partir de um determinado índice
- `length()` // devolve o número de caracteres da String

Exemplo de utilização:

```
StringBuffer a = new StringBuffer("Viva" + " o");
a.append(" java");
```

2. Considere que pretende implementar uma mini agenda pessoal de endereços de e-mail.

   Para armazenar a informação de um contacto crie uma classe, muito simples, `Contacto` com duas variáveis de instância: `nome` e `e-mail`;

   Implemente os seguintes métodos nesta classe:

```
Contacto()
Contacto(String nome, String e-mail)
Contacto(Contacto outro)

public String getNome();
public String getEmail();

void setNome(String nome);

/* só altere o email se o parâmetro contiver uma e só uma vez o
   símbolo "@" e este símbolo não deve ser for o último da String */
public void setEmail(String email);

/* métodos que verificam se o parâmetro é igual à respectiva variável
   de instância */
public boolean isMyName(String nome);
public boolean isMyEmail(String email);

/* método que implementa a comparação de objectos Contacto */
public boolean equals(Contacto);

/* Método que devolve uma string com a representação interna do
```

```
objecto. Utilize a classe StringBuffer internamente para criar o valor
   a devolver */
public String toString();
```

3. Implemente agora uma `Agenda` num Array de contactos com um limite de 100 entradas. Na agenda podem existir nomes repetidos, mas não endereços de email repetidos. Pretende-se implementar os seguintes métodos:

```
Agenda();

void add(Contacto c); /* adiciona um contacto ao fim da lista */
void add(String nome, String email);

void remove(String email);

/* métodos  de procura */
boolean existe(Contacto c);
String getNome(String email);

/* devolve o número de emails associado a um determinado nome */
String numEmails(String nome);

/* altera o nome do contacto com o email passado como parâmetro */
void setNome(String nome, String email);

/* determina se duas agendas são iguais, ou seja quando contém o mesmo
conjunto de contactos, mas não necessariamente pela mesma ordem */
boolean equals(Object c) ;

/* devolve uma agenda ordenada por nome */
Agenda ordena();
```

Nota: para a comparação de nomes utilize o método `compareTo` da classe String. Este método compara strings e da seguinte forma:

```
int x = a.compareTo(b)

x negativo => a < b
x = 0 => a = b
x positivo => a > b
```