**MC302 – Programação Orientada a Objetos**
**Instituto de Computação – UNICAMP**
Profa.: Cecília Mary Fischer Rubira

12/03/2013

# LAB01-B: Objetos, Classes e Tipos Abstratos de Dados

---

**CONHECENDO UM POUCO MAIS SOBRE O AMBIENTE ECLIPSE E A LINGUAGEM JAVA:**

O **Eclipse** é um ambiente (IDE – Integrated Development Environment) para o desenvolvimento de programas, inicialmente desenvolvida pela IBM e hoje, disponibilizado como uma ferramenta *open source* pelo consórcio chamado "Eclipse.org". O **Workspace** é responsável por administrar os recursos do usuário que são organizados em um ou mais projetos. Em **File → Switch workspace** você poderá alterar a área de trabalho do seu **Eclipse**.

Os projetos em Java normalmente são organizados em Pacotes (**package**). Os pacotes são estruturas de diretórios utilizadas para organizar classes e interfaces. Eles fornecem um mecanismo para reutilização de software que ajuda a prevenir o conflito de nomes. A Sun Microsystems, criadora da liguagem Java, especificou uma convenção para nomes de pacotes. Cada nome de pacote deve iniciar com seu nome de domínio Internet, na ordem inversa, seguido do nome do pacote. Portanto, no nosso curso, iremos usar sempre: br.unicamp.ic.mc302.<nome do pacote>

**Vetores em Java:**

Vetores são coleções de objetos ou tipos primitivos. Os tipos devem ser conversíveis ao tipo em que foi declarado o vetor:

```
int[] vetor = new int[10];
```

Cada elemento do vetor é inicializado a um valor *default*, dependendo do tipo de dados (null, para objetos; 0, para int, long, short, byte, float, double; Unicode 0, para char e **false**, para boolean). Elementos podem ser recuperados a partir da posição 0:

```
int elemento_1 = vetor[0];
int elemento_2 = vetor[1];
```

Vetores podem ser inicializados no momento em que são criados. Exemplo:

```
String[] semana = {"Dom", "Seg", "Ter","Qua", "Qui", "Sex","Sab"};
```

Todo vetor em Java possui a propriedade **length** que informa o número de elementos que possui, extremamente útil em blocos de repetição:

```
for (int x = 0; x < vetor.length; x++) {
      vetor[x] = x*x;
}
```

Uma vez criados, vetores não podem ser redimensionados.

Java possui uma coleção de APIs (bibliotecas) padrão, organizadas em pacotes, que podem ser usadas para construir aplicações. O `ArrayList` faz parte do JavaCollections Framework (JCF). Esta classe é bastante útil pois permite criar vetores dinâmicos e ainda oferece métodos úteis para manipular o conteúdo do mesmo. Para se utilizar esta classe basta colocar no inicio do seu arquivo:

```
import java.util.ArrayList;
```

---

**INSTRUÇÕES INICIAIS:**

1. Obtenha o arquivo lab01-b.zip (disponível no ensino aberto);
2. Descompacte esse arquivo no seu diretório de trabalho (workspace do Eclipse);
3. Siga as instruções para a criação de um projeto Java no eclipse (disponível no ensino aberto).
4. Use um programa de edição de textos disponível (e.g., *Microsoft Word, LibreOffice Writer*) para documentar as alterações de código realizadas bem como o resultado das suas execuções;
5. Ao final do lab, produza o arquivo no formato pdf e suba no ensino aberto no seu portifólio, disponibilizando-o para os formadores.

---

**Pacote** *br.unicamp.ic.mc302.contador*:

1. Abra o arquivo Contador.java e o arquivo ExemploContador.java, que define a classe ExemploContador. Compile os dois arquivos e execute a classe ExemploContador.

2. Implemente um programa que, dada uma seqüência de caracteres, use a classe Contador para contar o número de A's, E's, I's, O's, e U's existente nessa seqüência. Abra o arquivo ContadorVogais.java e complemente

a implementação do método main( ). Esse método deve contar o número de cada tipo de vogal presente na frase do vetor de caracteres fraseExaminada. Use a declaração abaixo:

```
Contador[] contVogais = new Contador[5]; // um para cada vogal

for (int i = 0; i < fraseExaminada.length; i++) {
      // analisar cada índice do vetor de caracteres fraseExaminada[i] e,
      // se for o caso, incrementar o contador da respectiva vogal.
}
```

---

**Pacote** *br.unicamp.ic.mc302.circulo*:

1. Abra o arquivo Circulo.java e o arquivo TestaCirculo.java, que define a classe TestaCirculo. Compile os dois arquivos e execute a classe TestaCirculo.

2. Modifique a classe TestaCirculos para:

a) criar um vetor de 5 objetos Circulo;
b) imprimir os valores x, y, raio de cada objeto;
c) declare outra referência do tipo Circulo[ ];
d) copie a referência do primeiro vetor para o segundo;
e) imprima ambos os vetores;
f) crie um terceiro vetor;
g) copie os objetos do primeiro vetor para o terceiro;
h) altere os valores de raio para os objetos do primeiro vetor;
i) imprima os três vetores.

3. Modifique a visibilidade dos atributos x e y da classe Circulo. Em seguida, altere a operação main( ) da classe TestaCirculo para imprimir os atributos do Circulo diretamente. Qual é a sua conclusão?

---

**Pacote** *br.unicamp.ic.mc302.listaInts*:

1. O pacote acima que acompanha este lab é um pequeno exemplo de como se cria um `ArrayList` para guardar coleções de Inteiros. Abra o arquivo ListaInts.java e estude-o. Em seguida abra o arquivo TestaLista.java, que define a classe TestaLista. Compile os dois arquivos e execute a classe TestaLista.

2. O método toString da classe `ListaInts` faz uso da classe `StringBuilder` para construir uma representação textual da lista de inteiros. Você pode explicar a diferença entre as classes `String` e `StringBuilder`?

3. Em seguida, modifique a classe `ListaInts` para remover elementos da Lista (Dica: use o método remove(objeto) de `ArrayList` que remove a primeira ocorrência do objeto, se existir). Modifique e execute a classe TestaLista para validar sua implementação.

4. Implemente em Java uma classe chamada MinhaMatriz com operações para inicializar cada um dos seus elementos e para multiplicar 2 matrizes de quaisquer dimensões multiplicáveis. Para esta tarefa **utilize** o ArrayList na representação das matrizes e as operações definidas por esta classe para manusear os valores da mesma. Por fim crie um programa principal que instancia 2 matrizes de dimensões multiplicáveis e depois as multiplica.