Programação 1

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

`http://moodle.ua.pt/`

Aula 7

deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

- Estruturas de Dados (Tipos Compostos)
- Introdução
- Criação de novos tipos de dados
- Declaração de variáveis de novos tipos
- Cópia de variáveis tipo referência
- Exemplos

Introdução (1)

- Os exemplos de programas apresentados até aqui foram muito simples em termos de comunicação com o utilizador.
- Quando estamos perante problemas mais complexos, com mais dados de entrada, torna-se mais complicado a decomposição do problema em funções dado que apenas podemos devolver uma variável de tipo primitivo por função.
- Há problemas onde seria interessante adequar um tipo de dados à representação da informação envolvida.
- Em muitas situações práticas, precisamos de armazenar informação relacionada entre si, eventualmente de tipos diferentes, na mesma variável.

Introdução (2)

- Todas as linguagens de programação permitem que o programador defina tipos de dados particulares para adequar a representação da informação às condições concretas do problema.
- Estes tipos de dados são designados normalmente por **Estruturas de Dados, Tipos Compostos** ou **Registos.**
- Na linguagem JAVA podemos utilizar classes (`class`)para a construção de registos.
- Um registo é então um tipo de dados que pode conter campos de cada um dos tipos básicos (`int`,`double`, `char`, `boolean`, `...`), ou outros tipos compostos.

Tipos de dados

- Tipos primitivos:
  - aritméticos:
    - inteiros:
      `byte, short, int, long`
    - reais:
      `float, double`
    - caracter:
      `char`
  - booleanos:
    `boolean`
- Tipos referência:
  `class` (registos), `array`, ...

Criação de um novo tipo de dados

- Estrutura de um programa (relembrar):

```
inclusão de classes externas
public class Programa{
  public static void main (String[] args){
    declaração de constantes e variáveis
    sequências de instruções
  }
  funções desenvolvidas pelo programador
}
definição de tipos de dados (registos)
```

- Os novos tipos de dados são criados depois da definição da classe do programa, neste momento no mesmo ficheiro.

Criação de um novo tipo de dados


```
class nomeDoTipo
{
    tipo1 nomeDoCampo1;
    tipo2 nomeDoCampo2;
    ...
    tipon nomeDoCampoN;
}
```

- A `class` define um novo tipo de dados referência constituido por vários campos.
- A partir desta definição passa a existir um novo tipo de dados, sendo possível declarar variáveis deste novo tipo.
- O acesso a cada um dos campos faz-se através do nome do campo correspondente.

Exemplo


```
class Complexo
{
    double real;
    double imag;
}
```

- Para declarar variáveis deste novo tipo temos que utilizar o operador new:

```
Complexo num = new Complexo();
```

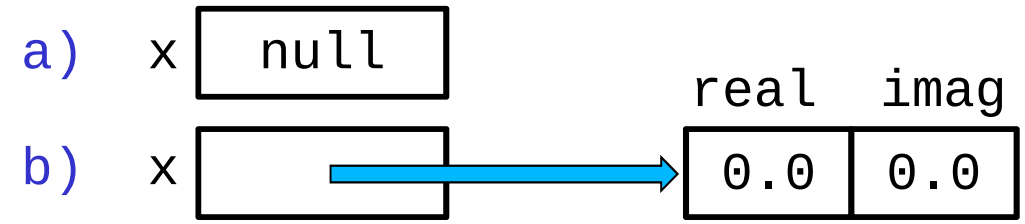
- num é uma variável que contem uma **referência** para um objeto criado do tipo Complexo.
- O operador new vai reservar espaço na memória do computador para a variável, o que permite a posterior utilização da mesma para armazenamento de dados.

Exemplo completo


```
public class registos1 {
  public static void main (String args[]){
    Complexo a, b;
    a = new Complexo();
    b = new Complexo();
    System.out.print("Parte real: ");
    a.real = nextDouble();
    System.out.print("Parte imaginaria: ");
    a.imag = nextDouble();
    ...
  }
}
class Complexo{
  double real, imag;
}
```

Declaração de uma variável


```
Complexo x; // a)
x = new Complexo(); // b)
```



- A declaração da variável x cria apenas uma referência para o que será mais tarde um número complexo.
- A invocação do operador new vai reservar espaço na memória do computador para o número complexo, ficando a variável x com o endereço onde esse "espaço" se encontra na memória.
- O operador new inicializa todos os campos da estrutura com o valor "0" (dependendo do tipo de dados do campo).
- A partir deste momento, o número complexo pode ser manipulado através da variável x.

- Tipos de **dados primitivos**:
  - a declaração da variável cria automáticamente a variável, reservando espaço em memória;
  - a variável é sempre passada por valor às funções como argumento.
- Tipos de **dados referência**:
  - a declaração da variável não cria de facto uma variável desse tipo, cria apenas uma referência;
  - a criação do objeto correspondente é feita com o operador `new`;
  - o objeto é sempre passado por referência como argumento às funções (veremos com mais detalhe na próxima aula...).

Cópia de variáveis tipo referência

- Atenção à cópia de uma variável tipo referência: é necessário distinguir a cópia do objeto da cópia da referência propriamente dita.
- Este é um dos erros frequentemente cometido pelos programadores.

```
Complexo x = new Complexo();
Complexo y = new Complexo();
x.real = 10;
x.imag = 20;
y = x; // estamos a copiar a referência e não o conteúdo
// Para copiar o conteúdo:
y.real = x.real; // cópia do campo real
y.imag = x.imag; // cópia do campo imag
```