

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



# Introdução (1)



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

- 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)



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

- 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, Registos ou Classes/Objetos**.
- Na linguagem JAVA podemos utilizar classes (`class`) para a construção de tipos compostos de dados.
- Uma classe é então um novo tipo de dados que pode conter campos de cada um dos tipos básicos (`int`, `double`, `char`, `boolean`, ...), ou outros tipos compostos.



# Tipos de dados



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

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



# Criação de um novo tipo de dados



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

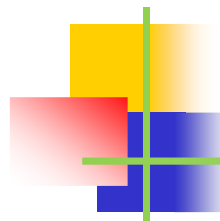
- Estrutura de um programa (relembrar):

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 compostos de dados são criados depois da definição da classe do programa, neste momento no mesmo ficheiro.



# Criação de um novo tipo de dados

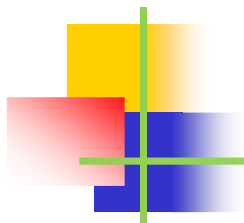


deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

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

- A `class` define um novo tipo de dados referência constituído 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



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

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

- Para declarar variáveis deste novo tipo (objetos) temos que utilizar o operador `new`:

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

- `num` é uma variável que contém 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

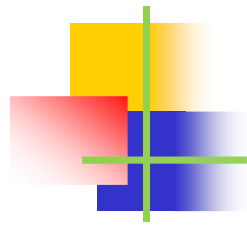


deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

```
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

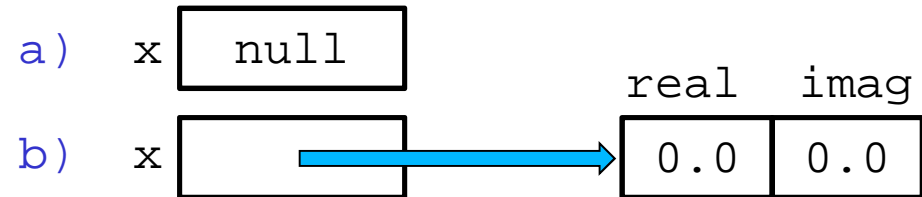


deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

```
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 automaticamente 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



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

- 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
```



# Arrays de registos



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

- Uma maneira de armazenar informação em aplicações reais consiste na utilização de sequências de registos, normalmente designadas por bases de dados.
- A declaração de arrays de registos é em em tudo semelhante à das sequências de tipos primitivos ou Strings, com a exceção que tem de ser decomposta em duas operações:
  - a primeira consiste em criar a sequência de referências para os futuros elementos do tipo registo;
  - a segunda consiste em criar os elementos propriamente ditos, seguindo a regra para a criação de variáveis do tipo registo.

# Exemplo (1)



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

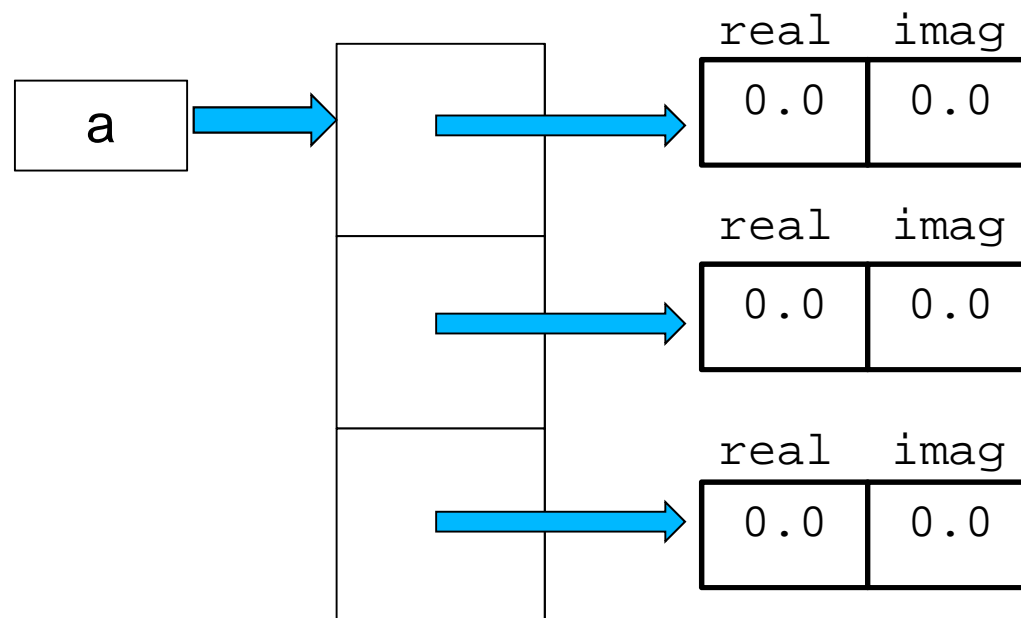
// Declaração de um array de números complexos

Complexo a[] = new Complexo[3]; // Declaração do array

a[0] = new Complexo(); // Alocação de espaço para pos. 0

a[1] = new Complexo(); // Alocação de espaço para pos. 1

a[2] = new Complexo(); // Alocação de espaço para pos. 2



## Exemplo (2)



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

```
// leitura de pontos até aparecer o (0, 0)
... main ...{
    Ponto2D pontos[] = new Ponto2D[10];
    Ponto2D p; int n = 0;
    do{
        System.out.println("Introduza um ponto:");
        p = lerPonto2D(); // aqui é criada uma nova referência
        if(p.x != 0 || p.y != 0){
            pontos[n] = p; // que depois é armazenada no array
            n++;
        }
    }while((p.x != 0 || p.y != 0) && n < pontos.length);
    imprimePontos(pontos, n);
}
```

## Exemplo (3)



deti

universidade de aveiro  
departamento de electrónica,  
telecomunicações e informática

```
public static Ponto2D lerPonto2D(){
    Ponto2D tmp = new Ponto2D();
    System.out.print("Coordenada x: ");
    tmp.x = sc.nextDouble();
    System.out.print("Coordenada y: ");
    tmp.y = sc.nextDouble();
    return tmp;}

public static void imprimePontos(Ponto2D a[], int n){
    for(int i = 0 ; i < n ; i++){
        System.out.printf("pto %d:(%.1f, %.1f)\n",
            i, a[i].x, a[i].y);
    }
}

class Ponto2D{
    double x, y;}
```