AULA TEÓRICA 1

Tema 1. Tipos de dados definidos pelo programador

- Introdução ao POO: noções básicos de classes e objectos.
- Criação de uma classe: atributos, construtores, comportamentos.
- Criação de objectos. Instanciação.

Classes e objectos

Boa parte de nosso entendimento e relacionamento com o mundo se dá através do conceito de objectos.

Principais características da programação orientada aos objectos:

- Identidade
- Classificação
- Polimorfismo
- Hereditariedade
- Encapsulamento.

Todos os objectos **tem um nome** que os representa. Ao mesmo tempo que cada objecto tem uma **identidade** e características próprias.

Reconhecemos categorias de objectos, ou seja, grupos de objectos que compartilham características comuns embora distintas em cada objecto, criarmos uma **classificação** para as coisas.

Polimorfismo se refere a nossa capacidade de nos reconhecer num objecto particular um outro mais geral .

Hereditariedade é um mecanismo de criarmos novos objectos a partir de outros que já existem.

Encapsulamento indica que podemos utilizar um objecto conhecendo apenas sua interface, isto é, sua aparência exterior .

Um **objecto** é uma combinação de dados ou atributos (variáveis) e acções ou comportamentos (métodos) logicamente relacionados.

Um objecto pode ser caracterizado por três componentes:

- identidade;
- atributos;
- comportamentos.

Em qualquer linguagem OO primeiro deve se definir a **classe** e depois objectos (um ou mais) que pertencem a esta classe, cada um deles com atributos próprios, mas com os mesmos comportamentos.

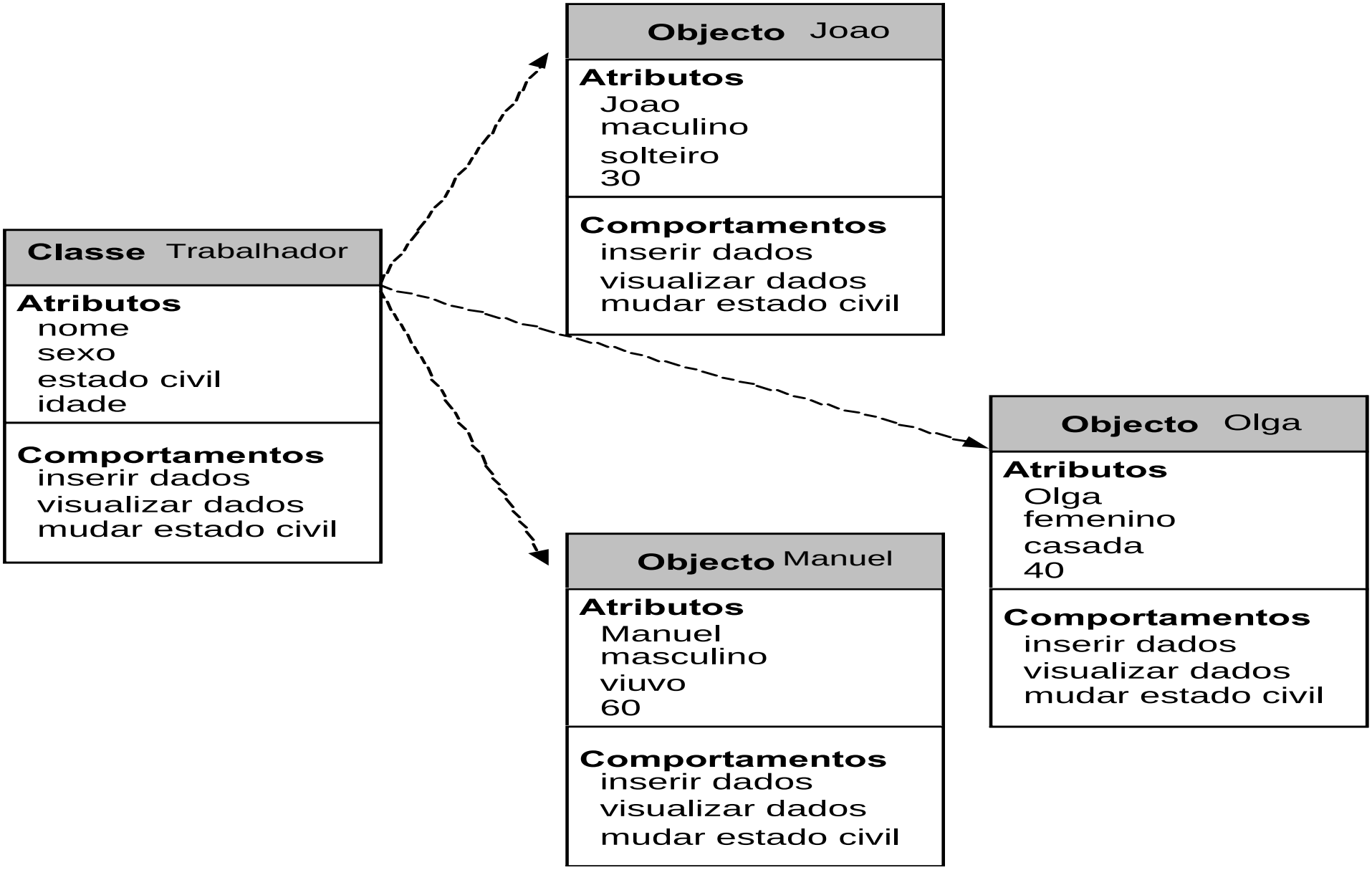
Podemos entender uma classe como um modelo ou como uma especificação para certos objectos.

Assim **a criação de uma classe** implica em definir um tipo de objecto em termos de seus atributos (variáveis que conterão os dados) e seus métodos (funções que manipulam tais dados).

Por exemplo, pretende-se introduzir dados sobre N trabalhadores e permitir a mudança de estado civil.

Para isso, será útil dispor de objectos que representam as propriedades e comportamentos dos trabalhadores. Como os objectos só podem ser criados a partir de uma classe, é preciso definir uma classe `Trabalhador`.

A relação entre uma classe e objectos dessa classe:



Atributos(Variáveis)

Na declaração de atributos utiliza-se a palavra reservada `private`, que serve para alterar as permissões de acesso a qualquer membro de uma classe, em vez de `public`.

Desta forma, indica-se ao compilador que estas variáveis só podem ser acedidas directamente pelos métodos (comportamentos) definidos na própria classe.

Assim, os atributos de um objecto criado a partir da classe `Trabalhador` mantêm-se **inacessíveis** a partir de qualquer outro objecto. Esta opção permite criar cada objecto como uma **entidade fechada** ou **encapsulada**.

Construtores

Cada classe deve ter pelo menos um construtor. Trata-se de **um tipo especial de método** utilizado apenas na criação e inicialização de objectos da classe. Distinguem-se dos restantes métodos por terem o **mesmo nome da classe** e por **não terem valor de retorno**, nem mesmo `void` .

Os construtores **não são chamados** como os outros métodos. Apenas a instrução de criação de objectos da classe os pode chamar. Estes métodos são usados muitas vezes para inicializar os atributos de um objecto.

Se nenhum construtor é definido para uma classe, o compilador cria um construtor *default* que não recebe nenhum argumento.

Boa prática fornecer um construtor para assegurar que cada objecto seja inicializado com valores significativos.

Nota:

1). A palavra `static` não é utilizada na definição da classe.

2). Não existe um método `main()` na definição da classe.

Comportamentos (métodos) de acesso aos atributos

Para além de construtor, podem ser considerados diversos comportamentos (métodos). Por vezes é útil permitir que objectos de outras classes tenham acesso às variáveis de instância (atributos) dos objectos, para consultar o seu valor ou para o alterar.

Existe um mecanismo para acesso controlado às suas variáveis de instância: **métodos de acesso**, que são geralmente simples.

Através deles devolve se o valor de um atributo (é comum utilizar a palavra **get** no início do nome do método) ou altera se o seu valor em função de um parâmetro (método com palavra **set** no início do nome).

Exemplo: Introduzir informação sobre código e idade de N trabalhadores e determinar a idade média de todos os trabalhadores. Utilizar POO.

```
import java.io.*;
public class Trabalhador
{ //Atributos
  private short codigo, idade;

  //Construtor. Recebe valores iniciais dos atributos (codigo e idade)
  public Trabalhador()throws IOException
  { System.out.println("Introduza o codigo (1111-9999): ");
    codigo = validarShort((short)1111, (short)9999);
    System.out.println("Idade (18-65): ");
    idade = validarShort((short)18,(short)65);
  }

  private short validarShort(short a, short b)throws IOException
  { short c; BufferedReader y =
                 new BufferedReader(new InputStreamReader(System.in));
    do
    { c = Short.parseShort(y.readLine());
      if(c<a || c>b)
        System.out.println("Valor invalido! Tente de novo");
     } while (c<a || c>b);
     return c;
  }
  public short getIdade()  { return idade; } //Devolve o valor de idade

  //devolve uma linha de informação sobre o trabalhador.
  public String toString()
  { return "Codigo\tIdade\n"+codigo+"\t"+idade; }
 }
```

Como se pode observar, a declaração dos atributos e a definição dos métodos é feita no interior da classe que começa com o cabeçalho:

```
public class Trabalhador
```

`Trabalhador` é o nome da classe que vai posteriormente permitir a criação de objectos desta classe. O código acima, uma vez guardado num ficheiro de nome `Trabalhador.java`, define um **novo tipo de dados.**

Criação de objectos. Instanciação

A criação de um objecto é o que chamamos **instanciação**. Instanciar significa criar uma instância da classe, isto é, um novo objecto que pode ser descrito através desta classe.

Enquanto uma classe é um modelo abstracto de um objecto, uma instância representa um objecto concreto desta classe. Do ponto de vista computacional, a instanciação corresponde a **alocação de memória** para armazenar informações sobre um certo objecto.

Depois de construir uma nova classe, já é possível escrever um programa que utiliza objectos criados a partir dessa classe. Para isso, o programador deve declarar os objectos e, posteriormente, reservar o espaço de memória necessário para os armazenar. A declaração de um objecto pode ser feita com uma declaração semelhante à das variáveis de tipos simples, para o caso do trabalhador:

```
Trabalhador trab;
```

Neste exemplo `trab` chamamos por **referência**, por permitir referenciar um objecto da classe `Trabalhador`, mas o seu valor, neste momento é indefinido.

A criação de objectos é feita através do operador **`new`**. Sintaxe:

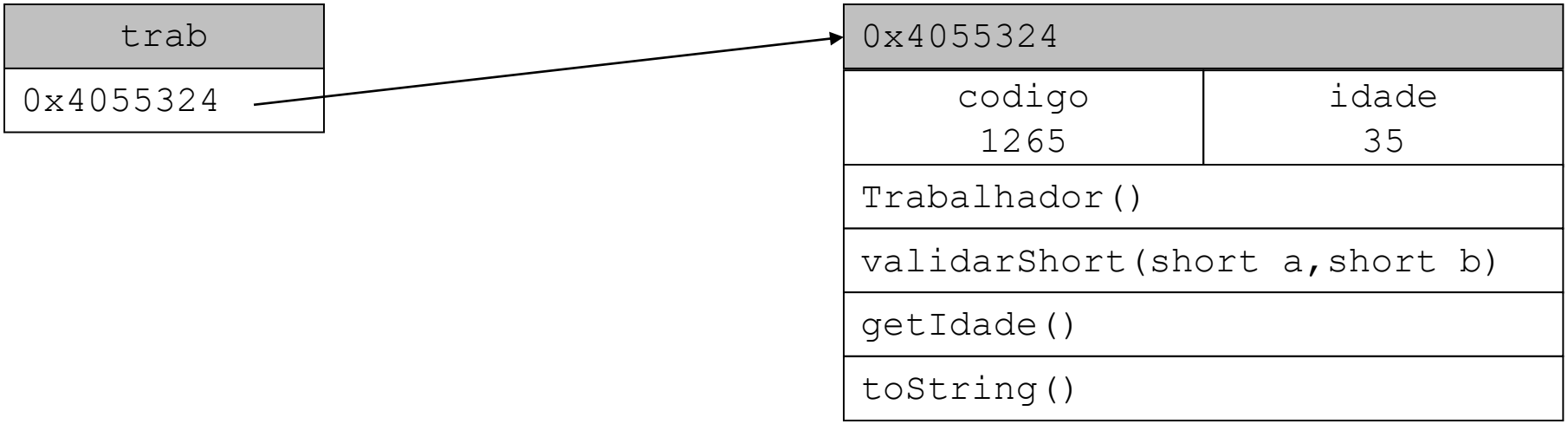
```
nomeDoObjecto = new ClasseDoObjecto(parâmetros);
```

O operador `new` verifica qual o espaço de memória necessário para armazenar o objecto (em função das características da classe) e reserva-o. O endereço respectivo é devolvido e armazenado na referência. Deste modo o acesso posterior ao objecto será possível através da referência.

O operador `new` encarrega-se de chamar o construtor da classe, permitindo a realização das operações de inicialização nele definidos.

Concretizando para o caso do trabalhador, a instrução seguinte permite criar um novo objecto da classe `Trabalhador`, guardar o seu endereço na referência `trab` e inicializar os seus atributos:

```
Trabalhador trab = new Trabalhador();
```



Em terminologia de POO, diz-se que o objecto é uma instância da classe, pelo que as suas variáveis (atributos) são **variáveis de instância.**

A classe `Trabalhador` define atributos `codigo` e `idade` para seus objectos. Para indicarmos qual destes atributos desejamos utilizar utilizamos um outro operador denominado **selector** indicado por um ponto como segue:

`nomeDoObjeto.nomeDoAtributo`

Usar um atributo pode significar uma de duas operações:
• consultar seu conteúdo (***get***), ou seja, ler ou obter seu valor e
• determinar seu conteúdo (***set***), isto é, escrever ou armazenar um certo valor neste atributo.

```
import java.io.*;
public class TrabalhadoresCriar
{ private int q;
  public  TrabalhadoresCriar() throws IOException
  { q = validQuant(); }
  private int validQuant() throws IOException
  { BufferedReader a =
          new BufferedReader(new InputStreamReader(System.in));
    do
    { System.out.println("Quantos sao trabalhadores ? ");
      q = Integer.parseInt(a.readLine());
      if( q <= 0)
        System.out.println("Valor invalido! Tente de novo");
    } while (q <= 0);
    return q; }
```

```
  //Devolve a quantidade
  public int getQuant()  { return q; }

  public float calcMedia()throws IOException
  { float soma = 0;
    for (int i = 1; i<= q; i++)
    { System.out.println("Dados do "+i+"-o trabalhador:");
      Trabalhador trab = new Trabalhador();
      System.out.println(trab);   //chamada do metodo toString()
      soma += trab.getIdade();
    }
    return Math.round(soma/q);
  }
}
```

==========

```
Import java.io.*;
public class Executavel
{ public static void main(String args[]) throws IOException
  { TrabalhadoresCriar t = new TrabalhadoresCriar();
    System.out.println ("idade media dos "+ t.getQuant()+
                        " trabalhadores="+ t.calcMedia());
  }
}
```