Programando a class Lamp

James Gosling
Criador da linguagem Java

Departamento de Informática FCT UNL (uso reservado ©)

Programando a classe Lamp em Java

- Programar em Java consiste em definir uma ou mais classes de objectos
- Definir uma classe em Java consiste em:
  - Escrever a sua especificação rigorosa, usando um editor de texto (integrado no IDE BlueJ)
- A linguagem Java, como qualquer linguagem de programação obedece a regras muito precisas, que é preciso conhecer com rigor
- Nos slides seguintes, vamos introduzir passo a passo os vários elementos presentes na definição de uma classe Java usando como exemplo a classe de objectos Lamp

Construção de um Programa

Texto do Programa
Lamp.java

```
public class Lamp{
   private boolean lampIsOn;
   public Lamp()    {
      lampIsOn = false;
   }
}
```

criado com um editor de texto

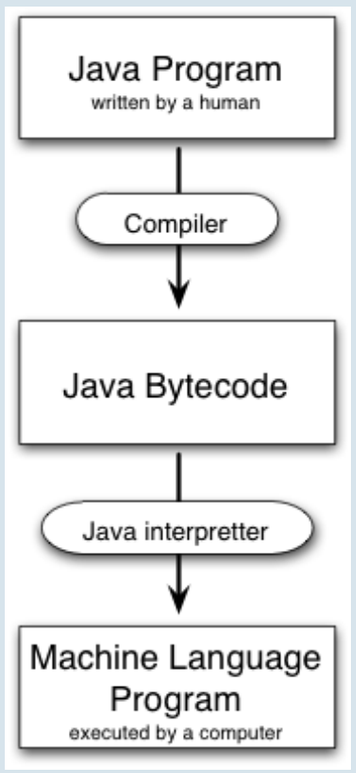
Compilador (javac)

o compilador **javac** é uma aplicação que traduz o texto do nosso programa em código máquina

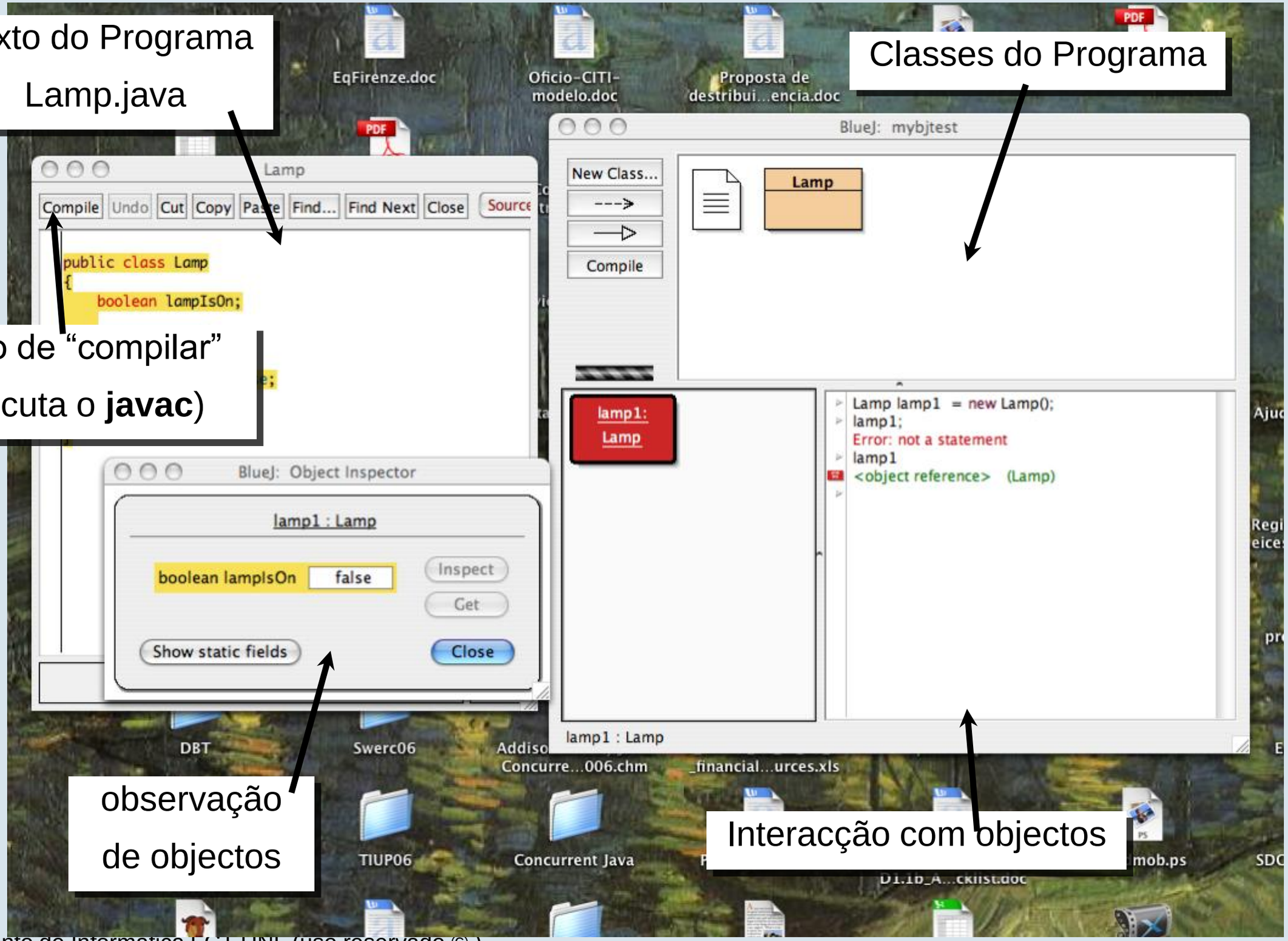
computador com uma máquina java instalada

Java Bytecode
Lamp.class

```
10010010010011101
00100111010010011
10100100111010010
01110100100111010
01001110100100111
01001001110100100
11101001001110100
10011101001001110
1001
```



O IDE BlueJ


Texto do Programa
Lamp.java
Classes do Programa
Botão de “compilar”
(executa o **javac**)
observação
de objectos
Interacção com objectos
Lamp
Compile
Undo
Cut
Copy
Find...
Find Next
Close
Source
public class Lamp
{
boolean lampIsOn;
BlueJ: mybjtest
New Class...
Compile
Lamp
lamp1:
Lamp
Lamp lamp1 = new Lamp();
lamp1;
Error: not a statement
lamp1
<object reference> (Lamp)
lamp1 : Lamp
BlueJ: Object Inspector
lamp1 : Lamp
boolean lampIsOn
false
Inspect
Get
Show static fields
Close
EqFirenze.doc
Oficio-CITI-modelo.doc
Proposta de destribui...encia.doc
DBT
Swerc06
Concurre...006.chm
_financial...urces.xls
TIUP06
Concurrent Java
mob.ps

Classe Lamp
(geradora de objectos “lâmpada”)

Aspecto geral da classe Lamp:

Nome da classe

```
public class Lamp {

   private boolean lampIsOn ;

   public Lamp() { lampIsOn = false; }

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Definição das operações dos objectos da classe Lamp

Classe Lamp
(geradora de objectos “lâmpada”)

A especificação de uma classe em Java tem sempre esta forma:

Nome da classe

```
public class ???? {
```

Aqui: Definição das variáveis e do construtor... veremos mais adiante

Aqui: Definição das operações (chamadas tecnicamente “métodos”) dos objectos da classe Lamp

```
}
```

Declaramos a classe como pública (**`public`**), para podermos aceder a ela livremente.

Classe Lamp
(geradora de objectos “lâmpada”)

Nome da classe

```
public class Lamp {
```

O nome da classe é um **identificador** qualquer, escolhido pelo programador (por si!)

Para identificador da classe use um nome (substantivo), iniciado por uma maiúscula

```
}
```

Identificadores (nomes) em Java

- Nos programas é necessário dar nomes a vários ingredientes (operações, classes, variáveis, etc)
- Os **identificadores** são escolhidos livremente pelo programador, e devem ser bons nomes para as coisas que pretendem identificar
- Os identificadores na linguagem Java
  - Podem incluir letras, dígitos e o carácter “_”
  - Não podem começar por um dígito
  - Não podem conter espaços
  - Não podem ser palavras já reservadas pela linguagem Java (por exemplo, **`class`**, **`int`**, etc.)

Identificadores (nomes) em Java

- Identificadores válidos:

    mySpace
    BladeRunner
    this_is_a_identifier
    agent007

-Sequências de caracteres que não são identificadores:

    0thesis (começa por um dígito)
    My My (contém um espaço em branco)
    class (palavra já reservada pela linguagem)
    What? (contém um caracter especial “?”)

Classe Lamp (definição dos métodos)

Cada declaração de método indica o programa associado a uma certa operação da interface do objecto

```
public class Lamp{

   public void on()   { … }

   public void off()  { … }

   public boolean status()  { … }
}
```

Métodos

Declaramos os métodos como públicos (**public**), para podermos usá-los livremente.

Classe Lamp (definição dos métodos)

Cada declaração de método indica o programa associado a uma certa operação da interface do objecto

O programa associado ao método `on` será escrito aqui entre as { ... }

```
public class Lamp{

   public void on()   { … }

   public void off()  { … }

   public boolean status()  { … }

}
```

Declaração do método `on`

Classe Lamp (definição dos métodos)

Cada declaração de método indica o programa associado a uma certa operação da interface do objecto

O programa associado ao método `off` será escrito aqui entre as { ... }

```
public class Lamp{

   public void on()   { … }

   public void off()  { … }

   public boolean status()  { … }

}
```

Declaração do método `off`

Classe Lamp (definição dos métodos)

Cada declaração de método indica o programa associado a uma certa operação da interface do objecto

O programa associado ao método `status` será escrito aqui entre as { ... }

```
public class Lamp {

   public void on()   { … }

   public void off()  { … }

   public boolean status()  { … }

}
```

Declaração do método `status`

Definição do método status

O estado da lâmpada, resultante da última operação modificadora efectuada, terá que ser guardado em **memória**

```
public class Lamp{

   public void on()   { … }

   public void off()  { … }

   public boolean status()  { … }

}
```

Programa associado ao método `status`

Vamos guardar o estado da lâmpada numa **variável**, à qual vamos dar o nome `lampIsOn`

Variáveis e Memória

- Uma variável em Java designa uma zona na memória do computador que pode guardar um valor de certo tipo
- As variáveis são referidas pelo seu nome (identificador)
- Quando um objecto é criado, devemos assumir que todas as suas variáveis contêm um valor indefinido.

- Para definir (ou substituir) o valor guardado numa variável, podemos usar o **comando de afectação**

```
public void on()  { lampIsOn = true;  }
public void off() { lampIsOn = false; }
```

afectação do valor **`true`** à variável `lampIsOn`

Variáveis e Memória

- Uma variável em Java é uma zona na memória do computador que pode guardar um valor de certo tipo
- As variáveis são referidas pelo seu nome (identificador)
- Quando um objecto é criado, devemos assumir que todas as suas variáveis contêm um valor indefinido.

- Para definir (ou substituir) o valor guardado numa variável, podemos usar o **comando de afectação**

```
public void on()  { lampIsOn = true;  }
public void off() { lampIsOn = false; }
```

afectação do valor **`false`** à variável `lampIsOn`

Memória de um Objecto

```
public class Lamp {
   private boolean lampIsOn ;
}
```

"memória" de cada objecto

A memória própria de cada objecto é especificada por um conjunto de **declarações de variáveis de instância**

Em Java, as declarações de variáveis de instância escrevem-se sempre na forma seguinte:

Acesso DescritorDeTipo Identificador ;
Acesso DescritorDeTipo Identificador = ValorInicial ;

Declaração de Variáveis

Modificador de acesso **`private`**. As variáveis de instância **não devem ser acessíveis fora da classe**.

```
public class Lamp {
    private boolean lampIsOn ;
}
```

DescritorDeTipo (**`boolean`**)

Identificador (`lampIsOn`)

A memória própria de cada objecto é especificada por um conjunto de **declarações de variáveis de instância**

Em Java, as declarações de variáveis de instância escrevem-se sempre na forma seguinte:

Acesso DescritorDeTipo Identificador ;

```
private boolean lampIsOn ;
```

Identificador na Declaração de Variável

O *Identificador* dá o nome à variável.
Esse nome é escolhido livremente pelo programador!

Identificador (`lampIsOn`)

```
public class Lamp {
    private boolean lampIsOn ;
}
```

A memória própria de cada objecto é especificada por um conjunto de **declarações de variáveis de instância**

Em Java, as declarações de variáveis de instância escrevem-se sempre na forma seguinte:

Acesso DescritorDeTipo Identificador ;

```
private boolean lampIsOn ;
```

Identificador na Declaração de Variável

O *Identificador* dá o nome à variável.
Esse nome é escolhido livremente pelo programador!

Identificador (`lampIsOn`)

```
public class Lamp {

    private boolean lampIsOn ;

}
```

Para nome da variável use um nome (substantivo), iniciado por uma minúscula

Em Java, as declarações de variáveis de instância escrevem-se sempre na forma seguinte:

Acesso DescritorDeTipo Identificador ;

```
private boolean lampIsOn ;
```

Descritor de Tipo na Declaração de Variável

DescritorDeTipo (**boolean**)

O *DescritorDeTipo* indica que valores podem ser guardados na variável

```
public class Lamp {
    private boolean lampIsOn ;
}
```

A memória própria de cada objecto é especificada por um conjunto de **declarações de variáveis de instância**

Em Java, as declarações de variáveis de instância escrevem-se sempre na forma seguinte:

Acesso DescritorDeTipo Identificador ;

```
private boolean lampIsOn ;
```

Tipos de dados em Java

- A linguagem Java suporta muitos **tipos de dados**.
- Para já, vamos conhecer os tipos **`boolean`** e **`int`** :
- O tipo **`boolean`**

  É o tipo dos valores lógicos (booleanos)

  Só existem dois valores de tipo **`boolean`**

  `false`

  `true`
- O tipo **`int`**

  É o tipo dos naturais positivos e negativos (inteiros)

  Existem muitos valores inteiros :

  `0`, **`1`**, **`-10`**, **`1000`**, etc..

  Em Java, os inteiros são limitados ao que é possível representar em 4 bytes **`[-2.147.483.648`**, **`2.147.483.647]`**.

Definição do método status

O estado da lâmpada, resultante da última operação modificadora efectuada é guardado na **variável** `lampIsOn`

```
public class Lamp {

   private boolean lampIsOn ;

   ...

   public void on()   { ... }

   public void off()  { ... }

   public boolean status()  { return lampIsOn; }

}
```

Programa associado ao método `status`

O comando "return `lampIsOn`" devolve o valor da variável `lampIsOn` como resultado do método `status`

O comando “return” return Expressão

- Para devolver o resultado de uma operação de um objecto é necessário usar o **comando `return`**
- A forma geral do comando **`return`** é

```
return Expressão
```

`Expressão` é uma expressão que indica o valor a devolver como resultado do método

- Depois de executar o comando

```
return Expressão
```

o programa no corpo do método **termina**, devolvendo o valor retornado ao utilizador.

Definição do método status

DescritorDeTipo (**`boolean`**)

Indica que o método devolve um valor booleano

( ) vazio, indica que o método não recebe dados de entrada

```
public boolean status() { return lampIsOn; }
```

Identificador (`status`)

Indica o nome do método

Corpo (**`return`** `lampIsOn;`)

Define o **programa que implementa o método**:

```
return lampIsOn;
```

Definição do método on

DescritorDeTipo (**`void`**)

Indica que o método não devolve nenhum resultado

( ) vazio, indica que o método não recebe dados de entrada

```
public void on() { lampIsOn = true; }
```

Identificador (`on`)

Indica o nome do método

Corpo (`lampIsOn = true;`)

Define o **programa que implementa o método**:

```
lampIsOn = true;
```

Definição do método off

DescritorDeTipo (**`void`**)

Indica que o método não devolve nenhum resultado

( ) vazio, indica que o método não recebe dados de entrada

```
public void off() { lampIsOn = false; }
```

Identificador (`off`)

Indica o nome do método

Corpo (`lampIsOn = false;`)

Define o **programa que implementa o método**:

```
lampIsOn = false;
```

O comando “afectação” VariableId = Expressão

- Uma variável em Java é uma zona na memória do computador, que pode guardar um valor de certo tipo
- Para definir (ou substituir) o valor guardado numa variável é necessário usar o **comando de afectação**
- A forma geral do comando afectação é

```
VariableId = Expressão
```

- `VariableId` é o identificador da variável a afectar
- `Expressão` é uma expressão que indica qual o valor a afectar à variável
- Depois da afectação ser executada, o valor previamente guardado na variável **perde-se para sempre**

Definição dos métodos

Em Java, as declarações de métodos escrevem-se sempre na forma seguinte:
Acesso DescritorDeTipo Identificador (*parâmetros*) { *Corpo* }

```
public class Lamp {

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Definição dos métodos

Em Java, as declarações de métodos escrevem-se sempre na forma seguinte:
Acesso DescritorDeTipo Identificador (*parâmetros*) { *Corpo* }

```
public class Lamp {

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Corpo (`lampIsOn = true;`)

DescritorDeTipo (**`boolean`**)

Identificador (`status`)

Definição dos métodos

Em Java, as declarações de métodos escrevem-se sempre na forma seguinte:
Acesso DescritorDeTipo Identificador (*parâmetros*) { *Corpo* }

```
public class Lamp {

    public void on() { lampIsOn = true; }

    public void off() { lampIsOn = false; }

    public boolean status() { return lampIsOn; }
}
```

DescritorDeTipo (**`void`**)

Identificador (`off`)

Corpo (**`return`** `lampIsOn;`)

Construtor da classe Lamp

Todas as classes têm que definir ainda um **construtor**

construtor de cada objecto da classe `Lamp`

```
public class Lamp {

   private boolean lampIsOn ;

   public Lamp() { lampIsOn = false; }

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Em geral, a função do **construtor** é definir os valores iniciais das variáveis de instância de cada objecto.

Construtor da classe Lamp

```
public class Lamp {

   public Lamp() { lampIsOn = false; }

}
```

construtor de cada objecto da classe `Lamp`

O construtor define qual é o valor inicial de todas as variáveis de instância do objecto.
Neste caso, define-se que o valor inicial da variável `lampIsOn` é o valor booleano **`false`**.

Construtor da classe Lamp

```
public class Lamp {

   public Lamp() { lampIsOn = false; }

}
```

construtor de cada objecto da classe `Lamp`

O construtor escreve-se:

Acesso `NomeDaClasse`() {. . .
afectação de valores iniciais
... }

Construtor da classe Lamp

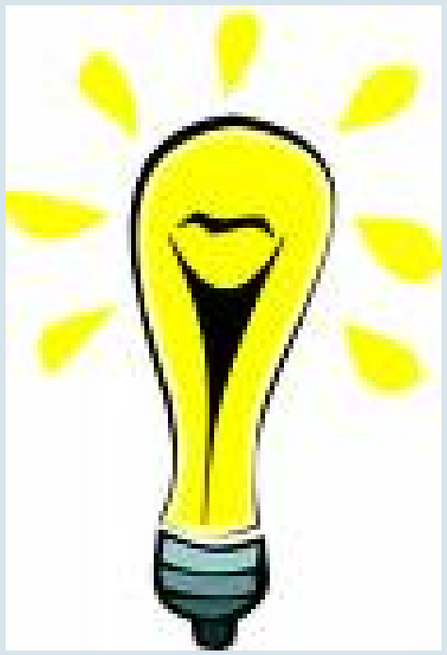
```
public class Lamp {

    public Lamp() { lampIsOn = false; }

}
```

O construtor escreve-se (neste exemplo):

```
public Lamp() { lampIsOn = false; }
```

Define o valor inicial de variável `lampIsOn` de **cada objecto** como sendo o valor booleano **`false`**

A classe Lamp (Resumo)



Departamento de Informática FCT UNL (uso reservado ©)

A Classe Lamp (Resumo)

```
public class Lamp {
   private boolean lampIsOn ;
   public Lamp() { lampIsOn = false; }
   public void on() { lampIsOn = true; }
   public void off() { lampIsOn = false; }
   public boolean status() { return lampIsOn; }
}
```

"memória" de cada objecto

A Classe Lamp (Resumo)

Em Java, a "memória" de cada objecto consiste num certo conjunto de variáveis

"memória" de cada objecto

```
public class Lamp {
   private boolean lampIsOn ;
   public Lamp() { lampIsOn = false; }
   public void on() { lampIsOn = true; }
   public void off() { lampIsOn = false; }
   public boolean status() { return lampIsOn; }
}
```

Neste exemplo, a memória é a variável `lampIsOn` que guarda o estado da lâmpada (**`true`** : acesa; **`false`** : apagada)

A Classe Lamp (Resumo)

A definição dos métodos associa **pequenos programas** ao nome de cada operação da interface dos objectos

```
public class Lamp {

   private boolean lampIsOn ;

   public Lamp() { lampIsOn = false; }

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }
}
```

Métodos

Os métodos `on` e `off` alteram o estado da lâmpada, enquanto o método `status` apenas consulta o estado da lâmpada

A Classe Lamp (Resumo)

Todas as classes têm que definir ainda um **construtor**

construtor de cada objecto

```
public class Lamp {

   private boolean lampIsOn ;

   public Lamp() { lampIsOn = false; }

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Em geral, a função do **construtor** é definir os valores iniciais das variáveis de cada objecto.

A Classe Lamp

```
public class Lamp {
   private boolean lampIsOn ;
   public Lamp() { lampIsOn = false; }
   public void on() { lampIsOn = true; }
   public void off() { lampIsOn = false; }
   public boolean status() { return lampIsOn; }
}
```

A Classe Lamp

Verifique cuidadosamente se compreendeu a finalidade de todos os elementos desta definição.

```
public class Lamp {

   private boolean lampIsOn ;

   public Lamp() { lampIsOn = false; }

   public void on() { lampIsOn = true; }

   public void off() { lampIsOn = false; }

   public boolean status() { return lampIsOn; }

}
```

Se tem alguma dúvida, procure esclarecê-la já e agora!

Forma das classes Java

Classes Java (Resumo da Forma Geral)

nome da classe

```
public class classNameId {

   private tipo varId;

   private tipo varId;

   ...

   public classNameId() { comandos }

   public tipo methodId() { comandos }

   public tipo methodId() { comandos }

   ...

}
```

declaração de variáveis

Classes Java (Resumo da Forma Geral)

```
public class classNameId {
    private tipo varId;
    private tipo varId;
    ...
    public classNameId()  {  comandos  }
    public tipo methodId()  {  comandos  }
    public tipo methodId()  {  comandos  }
    ...
}
```

construtor

métodos