Programação com Objectos

Fernanda Barbosa

Fernando Birra

Luís Caires

Armanda Rodrigues

Licenciatura em Engenharia Informática FCT UNL

Como é constituído o software?

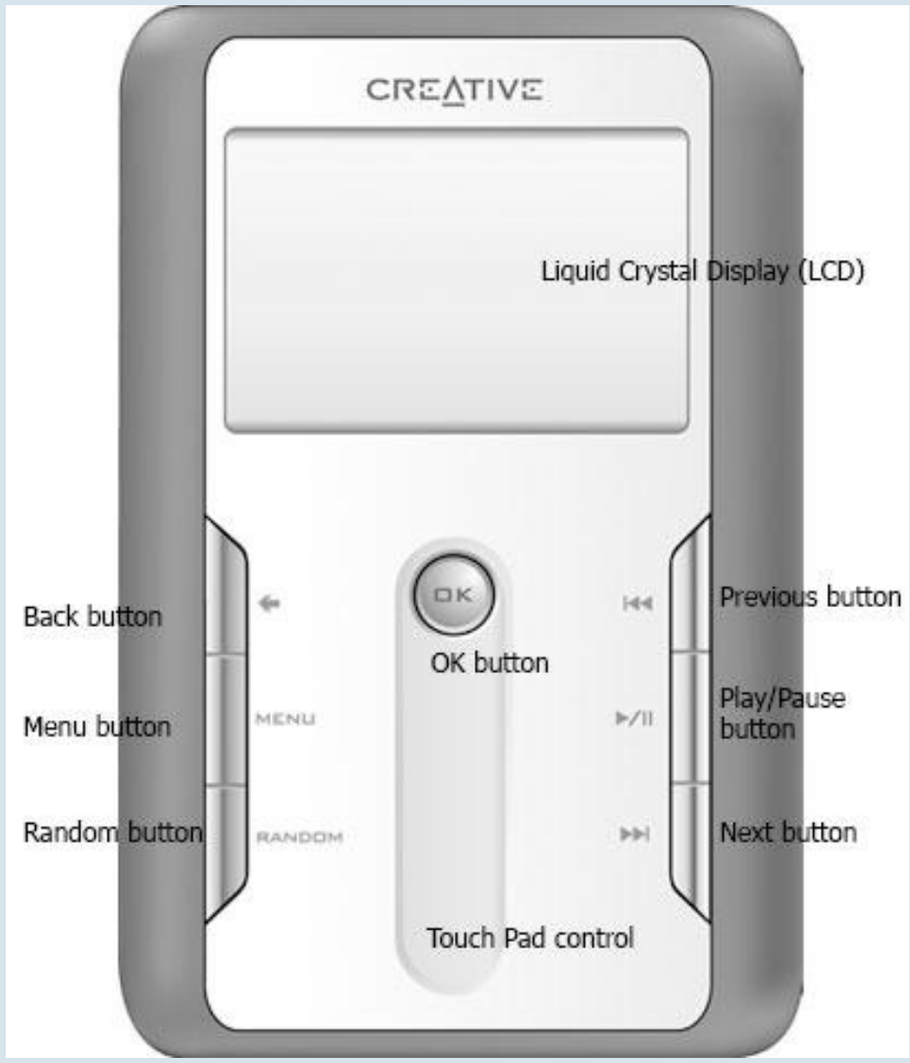
- As aplicações informáticas (software) são, em geral, sistemas bastante complexos
- Um programa descreve uma tarefa particular
  - ordenar uma lista de números
  - resolver um sistema de equações
  - visualizar uma fotografia
  - ...
- Uma aplicação realiza certas funcionalidades
  - edição de texto
  - sistema de base de dados
  - editor gráfico

Objectos

- Na sua maioria, os sistemas de software são organizados com base em componentes elementares, chamados **objectos**.
- No mundo real, um objecto é uma entidade física autónoma que desempenha uma determinada finalidade:
  - caneta
  - terminal multibanco
  - televisor
  - leitor MP3
  - automóvel

Objectos Físicos (analogia)

- Qualquer objecto físico suporta um conjunto de operações particulares, que lhe permitem ser utilizado pelo utilizador interessado.

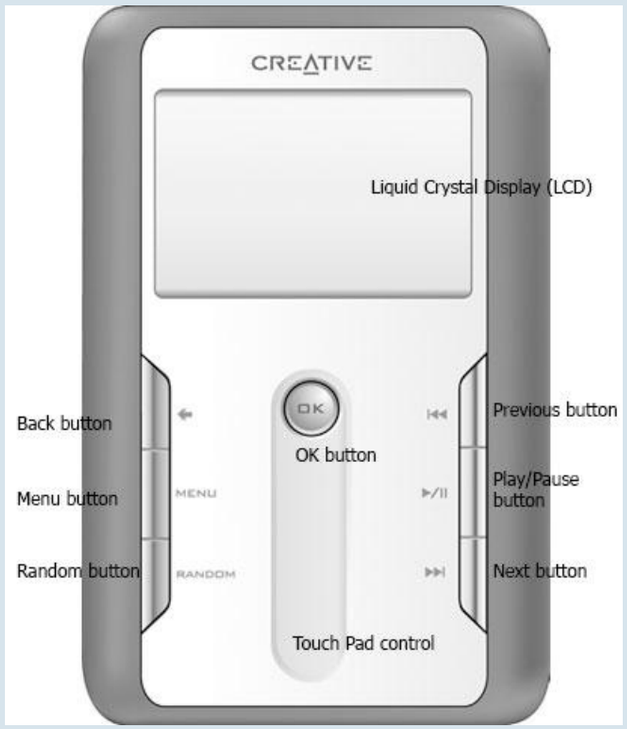


Objectos (Programação)

- Actualmente, os sistemas de software são organizados com base em componentes elementares, chamados **objectos**.
- No **mundo virtual** da programação, um objecto é uma entidade autónoma que desempenha uma determinada finalidade:
  - ficheiro
  - contador
  - conta bancária
  - agenda electrónica
  - base de dados

Interface do MP3

- Os utilizadores interagem com os objectos através da sua interface.
- Tecnicamente, chama-se **interface** à lista das operações suportadas por um objecto :



Interface do MP3:

- Previous
- Play
- Next
- Back
- Ok
- Playing?
- Memory?

Interface do MP3

- Alguma das operações são acções que podem ser aplicadas ao objecto. Por exemplo,
  - **Next**, muda para a música seguinte;
  - **Stop**, para a música corrente;
- Outras operações permitem apenas consultar o estado do aparelho;
  - **Playing?**, mostra no écran a música corrente;
  - **Memory?**, mostra no écran a memória livre;
- Em geral, todos os objectos apresentam estes dois tipos de operações na sua interface:
  - Operações Modificadoras (*mutators*)
  - Operações de Consulta (*accessors*)

Interface do Auto Rádio

- Trabalho na aula ...

Interface da Conta Bancária

- Trabalho na aula ...

O meu primeiro objecto (de software)

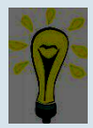
- Vamos definir um objecto (de software) que representa uma lâmpada.
- Queremos que a nossa lâmpada apresente as seguintes três operações na sua interface:

on()

coloca a lâmpada no estado “aceso”

off()

coloca a lâmpada no estado “apagado”



status()

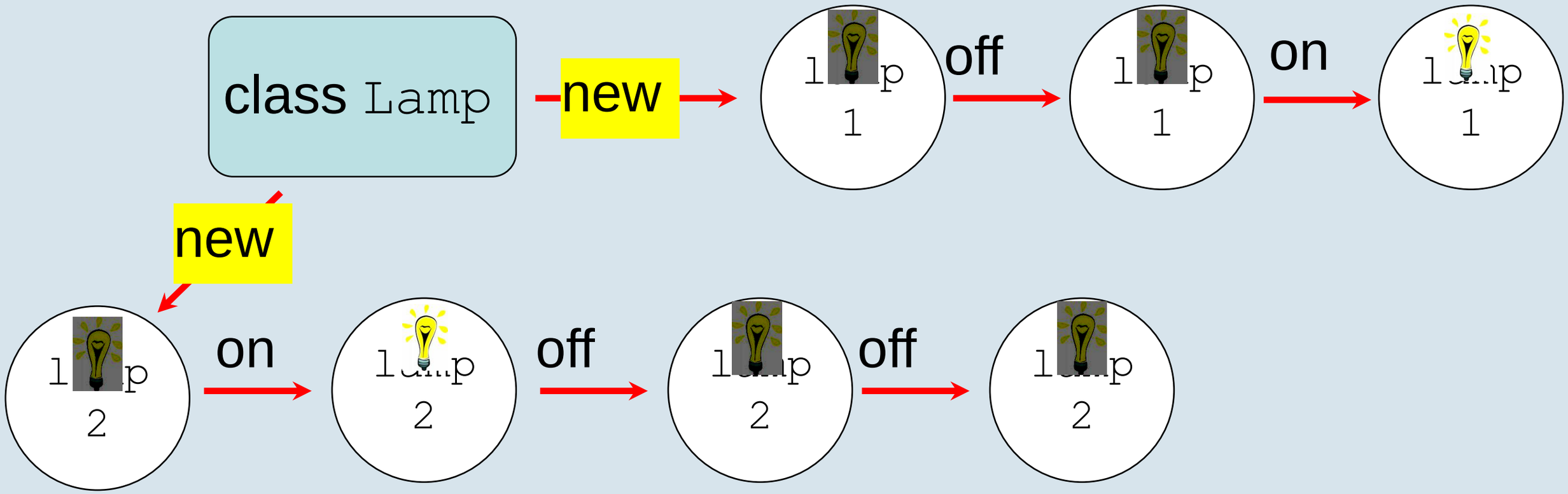
interroga à lâmpada qual é o seu estado corrente

- Nota: as operações on e off são modificadores, status é de consulta.

A “vida” de um objecto

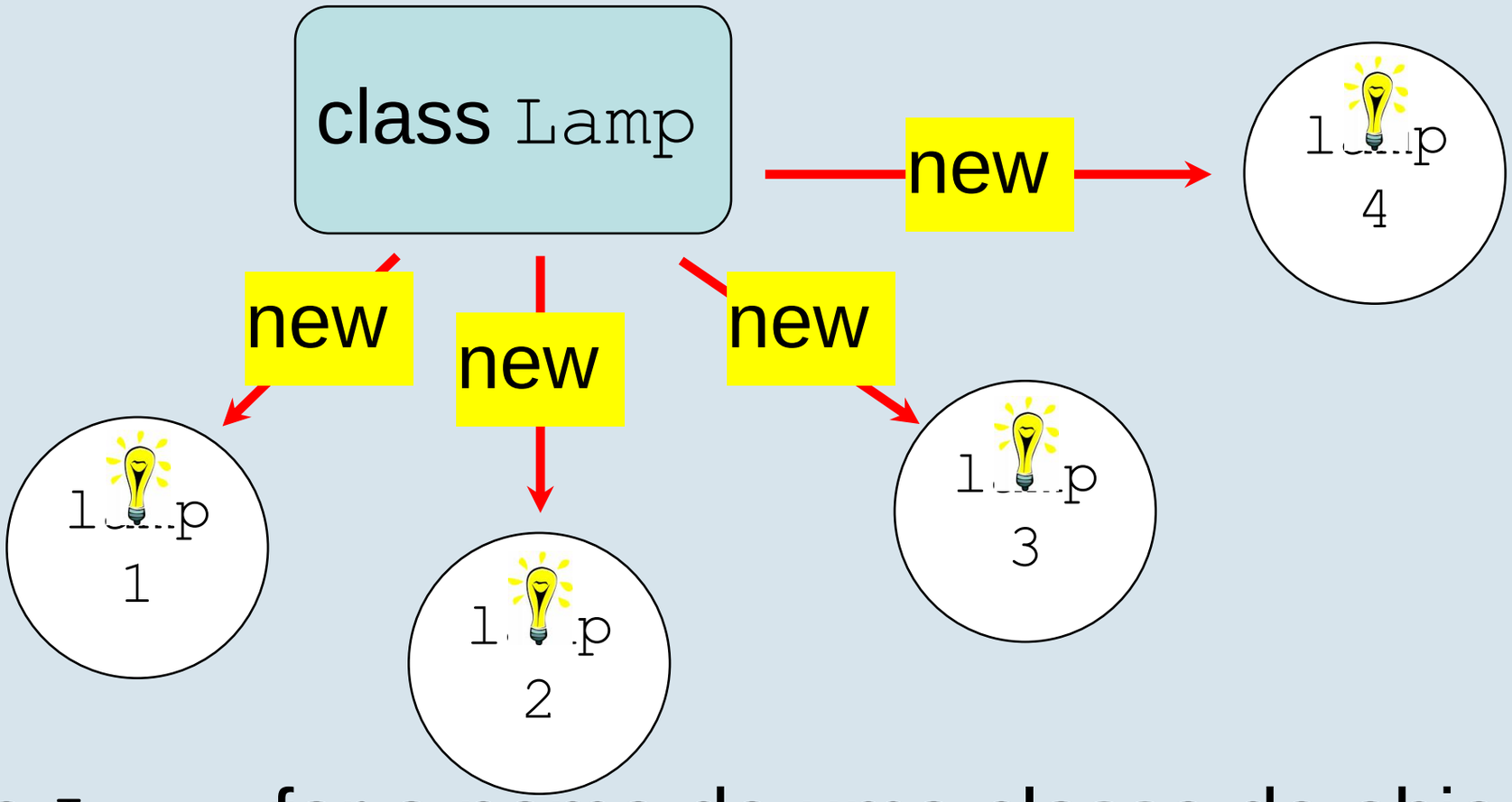
- Como são fabricados os objectos de software?
  - Os objectos são criados por componentes de software chamados “classes” (veremos adiante)
  - Classes funcionam como “fábricas” de objectos
  - Todos os objectos criados por uma classe são semelhantes à nascença:
    - apresentam a mesma interface
    - executam as mesmas operações
    - podem ter evoluções diferentes
    - as operações de um não “mexem” com o outro

A “vida” de um objecto



- Cada objecto tem a sua **vida própria**
- Cada objecto tem uma **identidade própria**:
  - Que é um código único, gerado pelo sistema
- Cada objecto **evolui autonomamente**

A “vida” de um objecto



- Em Java, se `Lamp` for o nome de uma classe de objectos, então a expressão new `Lamp`() representa a criação de um novo objecto da classe `Lamp`:

  `Lamp lamp1` = new `Lamp`();

  “seja `lamp1` um novo objecto da classe `Lamp`”

Comportamento esperado da lâmpada

- Quando a lâmpada acabou de ser criada, encontra-se no estado “apagado”.

- Sempre que for executada a operação on, a lâmpada deve passar do estado em que está para o estado “aceso”
- Sempre que for executada a operação off, a lâmpada deve passar do estado em que está para o estado “não aceso”, ou “apagado”
- Sempre que for executada a operação status, a lâmpada deve indicar o estado em que está.

Comportamento esperado da lâmpada

```
Lamp lamp1 = new Lamp();
```

→ Criação de uma nova lâmpada

```
lamp1.status()
```

false (boolean)

```
lamp1.on();
lamp1.status()
```

true (boolean)

```
lamp1.on();
lamp1.status()
```

true (boolean)

```
lamp1.off();
lamp1.status()
```

false (boolean)

Comportamento esperado da lâmpada

```
Lamp lamp1 = new Lamp();
lamp1.status()
```

false (boolean)

```
lamp1.on();
lamp1.status()
```

true (boolean)

```
lamp1.on();
lamp1.status()
```

false (boolean)

Invocação da operação `status`

Resposta da operação `status`

Nota: o resultado devolvido pelas operações de consulta é apresentado a **verde**

Comportamento esperado da lâmpada

```
Lamp lamp1 = new Lamp();
lamp1.status()
```
false (boolean)
```
lamp1.on();
lamp1.status()
```
true (boolean)
```
lamp1.on();
lamp1.status()
```
true (boolean)
```
lamp1.off();
lamp1.status()
```
false (boolean)

Invocação da operação `on`

A operação `on` não tem resposta (limita-se a alterar o estado do objecto lâmpada)

Nota: as operações `on` e `off` são **modificadores**

Demo

- Interacção com objectos da classe `Lamp` usando o BlueJ

Definição de objectos em Java

- A programação em Java consiste, em larga medida, na definição de novos objectos.
- Cada objecto disponibiliza um conjunto de operações, o programador tem que definir com rigor o efeito de cada uma dessas operações
- Cada operação de um objecto é definida por um “pequeno programa” (em geral, com menos de 10 linhas de texto).
- No contexto da programação, esses pequenos programas, que definem as operações dos objectos, chamam-se **métodos**
- Para além dos métodos, cada objecto possui uma **memória própria privada**, que estes podem utilizar.

Definição de classes em Java

- A programação em Java consiste, em larga medida, na definição de novos objectos.
- No entanto, não é prático definir cada objecto de forma independente.
- Assim, a linguagem Java permite a definição de **classes**, e não de objectos directamente.
- Uma classe é uma fábrica inesgotável de objectos do mesmo tipo.
- Exemplo: uma vez definida a classe `Lamp`, podemos criar quantos objectos da classe `Lamp` diferentes quisermos;
  - `mylamp1` = new `Lamp`();
  - `mylamp2` = new `Lamp`();
  - ...

Comportamento esperado da lâmpada

```
Lamp mylamp1 = new Lamp();
Lamp mylamp2 = new Lamp();
mylamp1.status()
```
false
```
mylamp2.on();
mylamp1.status()
```
false
```
mylamp1.on();
mylamp2.status()
```
true
```
mylamp1.status()
```
true

As operações efectuadas entre objectos diferentes da mesma classe não interferem umas com as outras