CONCEITOS BÁSICOS DE ORIENTAÇÃO A OBJETOS

ACH 2003 — COMPUTAÇÃO ORIENTADA A OBJETOS

Daniel Cordeiro

Escola de Artes, Ciências e Humanidades | EACH | USP

O QUE É UM OBJETO?

Programação Orientada a Objetos
É um estilo de programação centrado no uso de *objetos* para o projeto e construção de aplicações.

O QUE É UM OBJETO?

Programação Orientada a Objetos
É um estilo de programação centrado no uso de *objetos* para o projeto e construção de aplicações.

Definição (apesar de não existir consenso)
Objeto é um componente de software que contém propriedades (estado) e os métodos (comportamento) necessários para tornar um tipo de dado útil.

O QUE É UM OBJETO?

Programação Orientada a Objetos
É um estilo de programação centrado no uso de *objetos* para o projeto e construção de aplicações.

Definição (apesar de não existir consenso)
Objeto é um componente de software que contém propriedades (estado) e os métodos (comportamento) necessários para tornar um tipo de dado útil.

> *OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late--binding of all things. It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I'm not aware of them.* — *Dr. Alan Kay*

OBJETO

- Dados + Métodos
- Os métodos modificam o estado interno do objeto e servem como mecanismo primário para comunicação entre objetos

Encapsulamento (de dados)
Mecanismo de linguagens de programação para restringir o acesso ao estado interno de um objeto, fazendo com que toda interação com ele seja realizada através de seus métodos.

Benefícios

Modularidade: o código fonte de um objeto pode ser escrito e mantido independentemente do código de outros objetos

Encapsulamento: ao interagir com objetos pelos seus métodos, os detalhes de sua implementação interna se mantêm ocultos para o mundo externo

Reutilização: se um objeto já foi definido, você pode usá-lo no seu programa

***Pluggability* e facilidade de depuração:** se um objeto em particular se mostrar problemático, você pode removê-lo e plugar um outro em seu lugar

O QUE É UMA CLASSE?

Classe
é o modelo (ou protótipo) a partir do qual objetos individuais serão criados. Uma instância é um objeto construído a partir de uma classe.

```
class Bicicleta {
    int cadência = 0;
    int velocidade = 0;
    int marcha = 1;

    void mudarCadência(int novoValor) {
         cadência = novoValor;
    }
    void mudarMarcha(int novoValor) {
         marcha = novoValor;
    }
    void acelerar(int incremento) {
         velocidade = velocidade + incremento;
    }
    void frear(int decremento) {
         velocidade = velocidade - decremento;
    }
}
```

```
int cadência = 0;
int velocidade = 0;
int marcha = 1;
```

A linguagem Java define os seguintes tipos de variáveis:

Variáveis de classe: (campos estáticos), definidos com o modificador `static`, que indica para o compilador que existe apenas uma cópia dessa variável, independentemente do número de vezes que a classe foi instanciada

Variáveis de Instância: (campos que não são estáticos) seus valores são únicos para cada instância de uma classe

Variáveis locais: variáveis temporárias que só existem no escopo de um método

Parâmetros: variáveis que armazenam os valores (objetos) passados na chamada a um método

TIPOS PRIMITIVOS DE DADOS

byte	8-bits $[-128; 127]$
short	16-bits $[-32.768; 32.767]$
int	32-bits $[-2^{31}; 2^{31} - 1]$
long	64-bits $[-2^{63}; 2^{63} - 1]$
float	ponto flutuante de precisão simples de 32-bits
double	ponto flutuante de precisão dupla de 64-bits
boolean	`true` ou `false`
char	1 caractere Unicode de 16-bits

Além de suporte especial para cadeias de caracteres. Ex: `"isso é uma String"` (java.lang.String)

OPERADORES

Operador	Precedência
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >=` instanceof
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= \%= \&= ^= |= <<= >>= >>>=`

Tabela 1: Operadores e suas precedências. Quanto mais no topo da tabela, maior a precedência do operador.

if–then–else

```
void brecar() {
    if (estáEmMovimento) {
        velocidadeAtual--;
    } else {
        System.err.println("A bicicleta já está parada!");
    }
}
```

switch

```
public static int getMonthNumber(String month) {
    int monthNumber = 0;

    if (month == null) {
        return monthNumber;
    }

    switch (month.toLowerCase()) {
        case "january":
            monthNumber = 1;
            break;

        /*          ...          */

        case "december":
            monthNumber = 12;
            break;
        default:
            monthNumber = 0;
            break;
    }
    return monthNumber;
}
```

WHILE E DO–WHILE

```java
class WhileDemo {
    public static void main(
                        String[] args){

        int count = 1;
        while (count < 11) {
            System.out.println(
              "Count is: " + count);
            count++;
        }
    }
}
```

```java
class DoWhileDemo {
    public static void main(
                        String[] args){

        int count = 1;
        do {
            System.out.println(
              "Count is: " + count);
            count++;
        } while (count < 11);
    }
}
```

FOR

```java
class ForDemo {
  public static void main(String[] args){
    for(int i=1; i<11; i++){
      System.out.println(
                 "Count is: " + i);
    }
  }
}
```

```java
class EnhancedForDemo {
  public static void main(
                       String[] args){
    int[] numbers =
          {1,2,3,4,5,6,7,8,9,10};
    for (int item : numbers) {
      System.out.println(
                   "Count is: " + item);
    }
  }
}
```

BIBLIOGRAFIA

- The Java™ Tutorials
  https://docs.oracle.com/javase/tutorial/java/concepts/index.html