


Laboratório de Engenharia de Software

INF1636 – PROGRAMAÇÃO ORIENTADA A OBJETOS

Departamento de Informática – PUC-Rio

Ivan Mathias Filho
ivan@inf.puc-rio.br

Programa – Capítulo 1




Laboratório de Engenharia de Software

- Conceitos de Orientação a Objetos
 - Classe
 - Objeto
 - Mensagem
- Orientação a Objetos em Java

© LES/PUC-Rio 2

Programa – Capítulo 1




Laboratório de Engenharia de Software

- **Conceitos de Orientação a Objetos**
 - Classe
 - Objeto
 - Mensagem
- Orientação a Objetos em Java

© LES/PUC-Rio

3

Extensibilidade em Dados



Laboratório de Engenharia de Software

- Todas as linguagens de programação atuais oferecem, além de tipos de dados primitivos, tais como **int**, **double**, **char** e **boolean**, a possibilidade de criação de tipos de dados do usuário.

```
typedef struct vetor {  
    double x,y;  
} Vet;
```
- No exemplo acima, o tipo **Vet** foi criado para representar, a partir do tipo primitivo **double**, vetores no \mathbf{R}^2 .

© LES/PUC-Rio

4

Extensibilidade em Operações



- Além das operações básicas sobre os tipos primitivos, as linguagens de programação, em geral, permitem a construção de novas operações, por meio de funções e procedimentos.

```
double prodInterno(Vet v1,Vet v2) {
    return v1.x*v2.x+v1.y*v2.y;
}
```

- No exemplo acima, a função **prodInterno** permite calcular o produto interno de dois vetores do \mathbf{R}^2 .

© LES/PUC-Rio

5

Problema



- Linguagens como **C** e **Pascal** permitem operar livremente sobre os elementos básicos de tipos não-primitivos;
- Isso pode levar a situações em que o valor de uma variável seja incompatível com o seu tipo.

```
typedef struct data {
    int dia,mes,ano;
} Data;

int main(void) {
    Data dtNasc;


    dtNasc.mes=100; // Inconsistente!!!
}
```

© LES/PUC-Rio

6

Laboratório de Engenharia de Software

Solução



- Esse problema pode ser solucionado por meio de um mecanismo sintático que bloqueie o livre acesso aos elementos básicos dos tipos não-primitivos;
- Desse modo, a manipulação de dados baseados em tipos não-primitivos só poderia ser feita por meio de operações definidas especialmente para esses tipos;
- Esse mecanismo é baseado no conceito matemático de Tipo Abstrato de Dados (ADT);
- Ele é chamado de **Classe**.

Uma classe é um tipo abstrato de dados possivelmente parcialmente implementado.


Bertrand Meyer
Object Oriented Software Construction

© LES/PUC-Rio

7

Laboratório de Engenharia de Software

Classe - Definição



- Outra definição, mais operacional, de classe é:

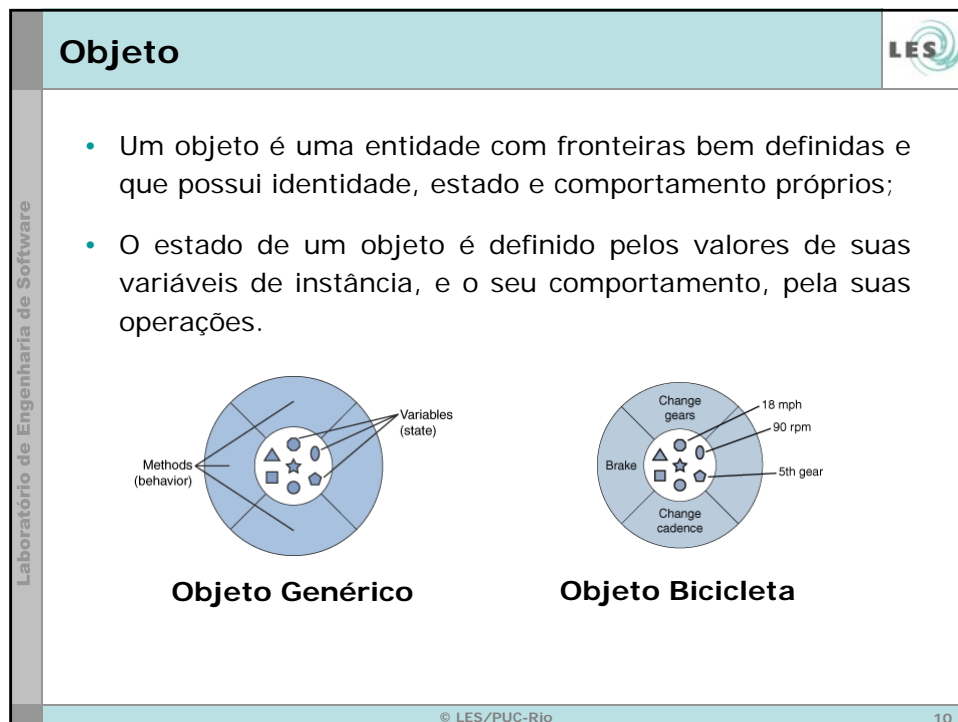
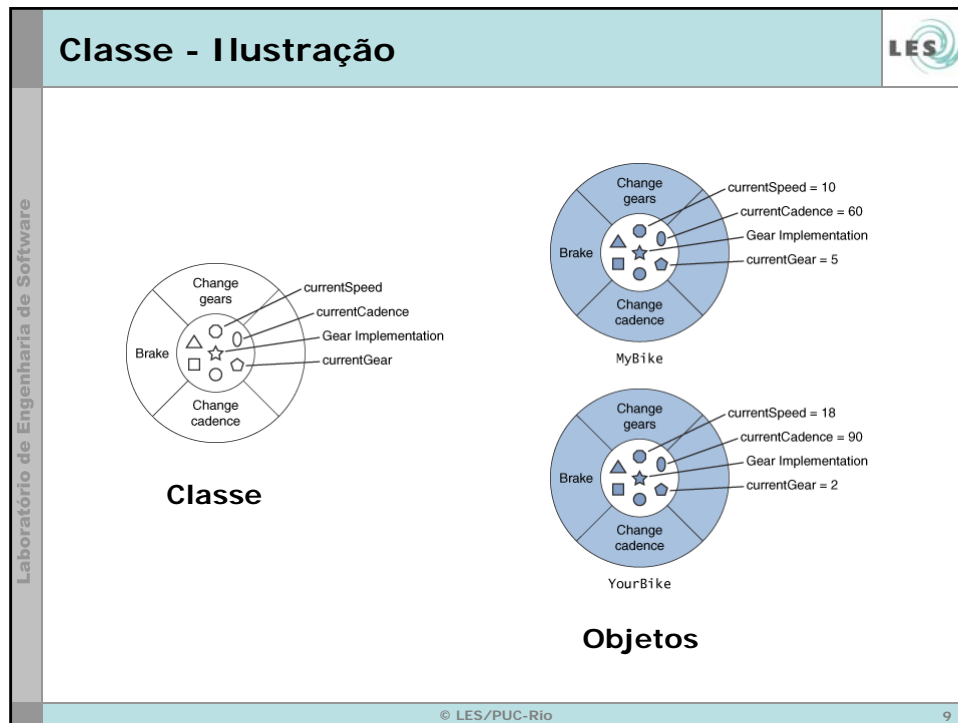
Uma classe é a descrição de um conjunto de objetos que possuem a mesma semântica e compartilham as mesmas propriedades (atributos, operações e relacionamentos).

Grady Booch
UML User Guide

- Nessa definição, uma classe é vista como um conjunto de objetos;
- Um objeto é uma instância de alguma classe.

© LES/PUC-Rio

8



Objetos



- Nesse modelo de objeto, as variáveis de instância são **mantidas** no seu centro, ou núcleo.
- As operações (métodos) rodeiam e escondem o núcleo de um objeto dos demais objetos existentes em um programa.
- O empacotamento da estrutura interna de um objeto é denominado **encapsulamento**.
- Tal modelo (um núcleo contendo variáveis de instância protegidas por operações) é considerado por muitos como a representação **ideal** de um objeto de software.

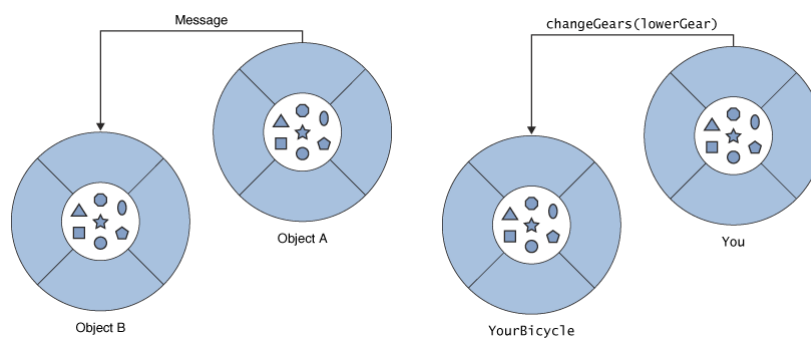
© LES/PUC-Rio

11

Mensagens




- Objetos interagem por meio de **mensagens**;
- Quando um objeto **A** deseja que um objeto **B** execute uma de suas operações, **A** envia uma mensagem para **B**;
- As informações passadas por meio de uma mensagem são os **parâmetros** da operação a ser executada.



Laboratório de Engenharia de Software

Programa – Capítulo 1




- Conceitos de Orientação a Objetos
 - Classe
 - Objeto
 - Mensagem
- **Orientação a Objetos em Java**

© LES/PUC-Rio

13

Laboratório de Engenharia de Software

Classe em Java – Exemplo



Class

declaration

Variables

Constructor

Methods

```
import java.util.*;

public class Stack {
    private List<Object> items;

    public Stack() {
        items = new ArrayList<Object>();
    }

    public void push(Object item) {
        items.add(item);
    }

    public Object pop() {
        if (items.size() == 0)
            throw new EmptyStackException();
        return items.remove(items.size() - 1);
    }

    public boolean isEmpty() {
        return items.isEmpty();
    }
}
```

© LES/PUC-Rio

14

Classes em Java – Estrutura Básica



- Uma classe Java possui a seguinte estrutura básica:

```
<modificadores> class <nome> {
    // variáveis
    <lista_modificadores> <tipo> <nome> [=<valor_inicial>;]

    // métodos
    <lista_modificadores> <tipo> <nome>([<lista_parametros>]) {
        <comandos>
    }
}
```

© LES/PUC-Rio

15

Classes em Java – Exemplo



```
public class Vetor {
    // variaveis
    private double x=3.0;
    private double y=4.0;

    // métodos
    public double modulo() {
        double a;

        a=Math.sqrt(Math.pow(x,2.0)+Math.pow(y,2.0));
        return a;
    }
}
```

© LES/PUC-Rio

16

Modificador de Acesso public



- Na definição de uma classe, o modificador **public** declara que a mesma pode ser usada por qualquer outra classe.
- A ausência do modificador **public** faz com que uma classe só possa ser usada por classes que pertençam ao mesmo pacote que a classe em questão;
- Na definição de uma variável de instância, o modificador **public** declara que ela pode ser livremente acessada por métodos externos à classe na qual foi definida.

```
public class Teste {
    private Vetor v;

    public void umMetodo()
    {
        v.x=5.0; // quebra do encapsulamento
                // indesejável!!!!
    }
}
```

© LES/PUC-Rio

17

Modificador de Acesso private



- Para não permitir a quebra do encapsulamento, devemos usar o modificador de acesso **private** na definição de variáveis de instância;
- Uma variável privada só pode ser acessada por métodos definidos na própria classe a qual ela pertence.

```
public class Teste {
    private Vetor v;

    public void umMetodo()
    {
        v.x=5.0; // quebra do encapsulamento
                // indesejável!!!!
    }
}
```

The field Vetor.x is not visible
Press 'F2' for focus

- Os métodos de uma classe são normalmente definidos com o modificador de acesso **public**.

© LES/PUC-Rio

18