

Classes e Objetos

Lilian Passos Scatalon
lpscatalon2@uem.br

Trabalhando com objetos

O programa do Unix `cal` imprime o calendário do mês atual no seguinte formato:

```
lilian@vostro:~$ cal
      Dezembro 2022
do se te  qu  qu se  sá
                1  2  3
 4   5   6   7   8   9 10
11  12  13  14  15  16 17
18  19  20  21  22  23 24
25  26  27  28  29  30 31
```

Como implementar um programa assim?

Trabalhando com objetos

Dezembro 2022						
do	se	te	qu	qu	se	sã
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Como representar as datas?

Imprimir as duas primeiras linhas

Como saber o mês atual?

Imprimir os dias do mês atual

Como saber em qual dia da semana cai o dia 01?

Como saber qual é o último dia do mês?

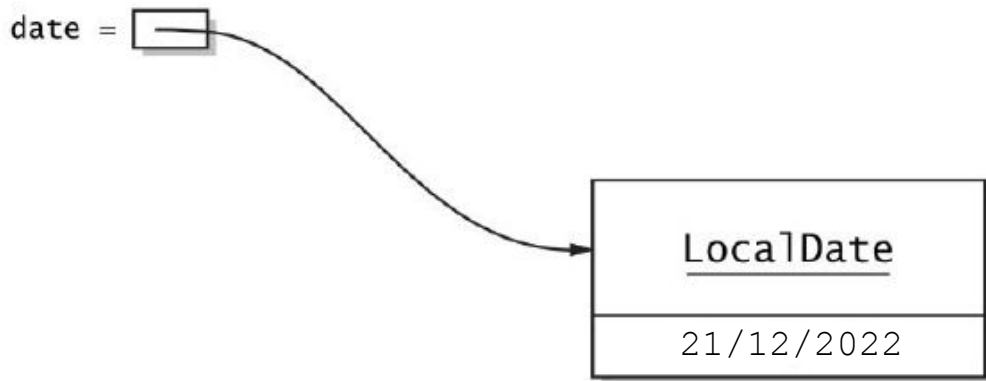
Trabalhando com objetos

Existe uma classe para datas na API Java, chamada **LocalDate**

Um objeto representa a data de hoje:

```
LocalDate data = LocalDate.now();
```

data: 21/12/2022



Trabalhando com objetos

Existe uma classe para datas na API Java, chamada **LocalDate**

Um objeto representa a data de hoje:

```
LocalDate data = LocalDate.now();
```

data: 21/12/2022

O mês atual é recuperado da data de hoje:

```
mes = data.getMonthValue()
```

mês: 12

Troca-se o dia atual pelo dia 01:

```
data = data.withDayOfMonth(1);
```

data: 01/12/2022

Trabalhando com objetos

Outro método retorna o dia da semana em que uma data cai:

```
DayOfWeek diaSemana = data.getDayOfWeek();
```

diaSemana: quinta-feira

Para a indentação do início do calendário, é preciso saber o valor numérico do dia da semana:

```
int valor = diaSemana.getValue();  
for(int i=1; i<valor; i++)  
    System.out.println("    ");
```

valor: 4 (segunda-feira é 1)

Trabalhando com objetos

A data é incrementada em um dia para gerar a próxima data:

```
data = data.plusDays(1);
```

data + 1 dia: 02/12/2022

Incrementando a data um dia por vez, dentro de um loop, são gerados todos os dias do mês

03/12/2022, 04/12/2022, ..., 30/12/2022, 31/12/2022, 01/01/2023

O mês da data é usado como condição de parada do loop:

```
while (data.getMonthValue() == mes) {  
    data = data.plusDays(1);  
    ...  
}
```


Trabalhando com objetos

É possível encadear chamadas de métodos:

```
int valor = dia.getDayOfWeek().getValue();
```



retorna um objeto de DayOfWeek



o método `getValue` é chamado a partir deste objeto retornado


```

import java.time.DayOfWeek;
import java.time.LocalDate;

public class Cal {
    public static void main(String[] args) {
        LocalDate date = LocalDate.now().withDayOfMonth(1);
        int month;
        if (args.length >= 2) {
            month = Integer.parseInt(args[0]);
            int year = Integer.parseInt(args[1]);
            date = LocalDate.of(year, month, 1);
        } else {
            month = date.getMonthValue();
        }

        System.out.println(" Mon Tue Wed Thu Fri Sat Sun");
        DayOfWeek weekday = date.getDayOfWeek();
        int value = weekday.getValue(); // 1 = Monday, ... 7 = Sunday
        for (int i = 1; i < value; i++)
            System.out.print(" ");
        while (date.getMonthValue() == month) {
            System.out.printf("%4d", date.getDayOfMonth());
            date = date.plusDays(1);
            if (date.getDayOfWeek().getValue() == 1)
                System.out.println();
        }
        if (date.getDayOfWeek().getValue() != 1)
            System.out.println();
    }
}

```

Objetos colaborando entre si

Neste caso, objetos que pertencem a classes pré-definidas

Como declarar novas classes?

Objetos e classes

Um programa orientado a objetos contém um modelo de alguma parte do mundo

Os objetos em Java modelam objetos do mundo real, que fazem parte de um domínio de problema, por exemplo:

- Palavras e parágrafos em um editor de texto
- Usuários e mensagens em uma rede social
- Personagens e monstros em um jogo
- Datas, compromissos, lembretes em uma agenda

Objetos e classes

Objetos podem ser agrupados em classes

Uma classe descreve, de modo abstrato, todos os objetos de um determinado tipo

Por exemplo, para modelar uma locadora de veículos

- Carro é uma classe ou um objeto?
- De que cor é um carro? Quão rápido pode andar? Onde está agora?

Classes e objetos

A palavra “carro”, de modo geral, se refere à classe carro

Para um carro em particular, “meu carro”, pode responder às questões anteriores

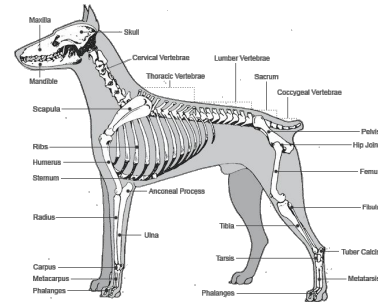
- O carro é preto, não anda muito rápido e está no estacionamento da UEM

Trata-se de um objeto, um exemplo particular de carro

Abstração

Representação de uma entidade que inclui apenas os atributos mais significativos

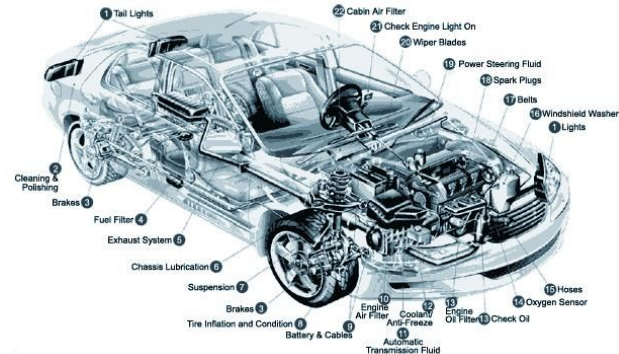
Envolve a omissão de detalhes de baixo nível, resultando em uma ideia mais simples, de alto nível



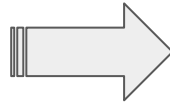
Abstração

Representação de uma entidade que inclui apenas os atributos mais significativos

Envolve a omissão de detalhes de baixo nível, resultando em uma ideia mais simples, de alto nível



Abstração



Classes e objetos

Uma classe **modela** as **características** e **comportamentos** que um conjunto de objetos têm em comum

Em uma locadora de veículos, o modelo de um carro pode ser

- Características: marca, modelo, ano, disponível
- Comportamentos: emprestar, devolver, verificar se está disponível

Classes e Objetos

Em POO, um programa consiste em objetos colaborando entre si

Objetos são instâncias de uma classe

- 21/12/2022, 23/01/2023, 06/02/2023 são instâncias de data

A classe é o modelo que define os objetos que pertencem a ela

- Uma data é composta por dia, mês e ano

Classe

Uma **classe** é um template/modelo que define o formato dos objetos que pertencem a ela

A classe é uma abstração lógica, um plano de como o objeto deve ser construído

Apenas quando um objeto da classe é criado que existe uma representação física na memória

```
public class Funcionario {  
    ...  
}
```

Classe

Uma **classe** é composta por:

- **atributos**: variáveis que representam as características
- **métodos**: subprogramas que implementam os comportamentos

```
class NomeClasse {  
    // declara atributos  
    tipo var1;  
    tipo var2;  
    // ...  
    tipo varN;  
  
    // declara métodos  
    tipo metodo1(parametros) {  
        // corpo do método  
    }  
    tipo metodo2(parametros) {  
        // corpo do método  
    }  
    // ...  
    tipo metodoN(parametros) {  
        // corpo do método  
    }  
}
```

Atributos

São as variáveis declaradas dentro de uma classe

Podem ser variáveis de instância ou de classe (`static`)

Variáveis de instância descrevem o estado de um objeto

```
public class Funcionario {  
    public String nome;  
    public double salario;  
    ...  
}
```

Cada instância da classe `Funcionario` tem essas duas variáveis

Métodos

Subprogramas que implementam as operações da classe

Podem ser **métodos de instância** ou de classe (`static`)

A classe pode incluir o método `main` se for o ponto de partida do programa, mas não é obrigatório

Métodos

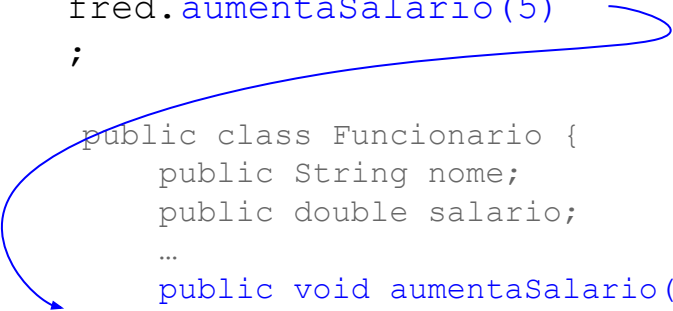
Métodos de instância são chamados/invocados a partir de uma referência de objeto

```
public class Funcionario {  
    public String nome;  
    public double salario;  
  
    public void aumentaSalario(double porcentagem) {  
        double aumento = salario * porcentagem / 100;  
        salario += aumento;  
    }  
    ...  
}
```

Métodos

Métodos de instância são chamados/invocados a partir de uma referência de objeto

```
fred.aumentaSalario(5);
```



```
public class Funcionario {  
    public String nome;  
    public double salario;  
    ...  
    public void aumentaSalario(double porcentagem) {  
        double aumento = salario * porcentagem / 100;  
        salario += aumento;  
    }  
    ...  
}
```

Métodos

Métodos de instância podem acessar/alterar variáveis de instância

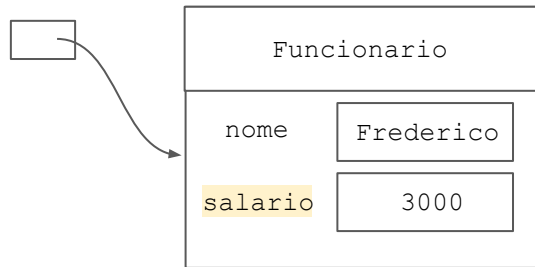
```
fred.aumentaSalario(5)
```

```
;
```

```
public class Funcionario {  
    public String nome;  
    public double salario;  
    ...
```

```
    public void aumentaSalario(double porcentagem) {  
        double aumento = salario * porcentagem / 100;  
        salario += aumento;  
    }  
    ...  
}
```

fred



Objeto

Objeto é uma **instância** da classe (uma variável do tipo classe)

Objetos são do **tipo de referência**, portanto a variável contém apenas uma referência (endereço)

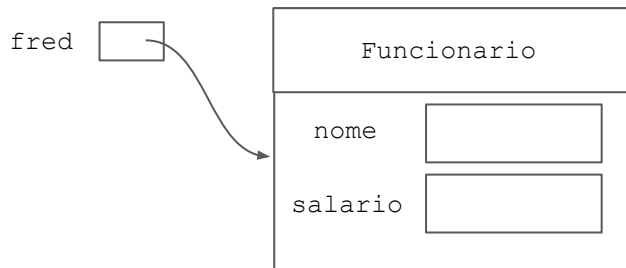
```
Funcionario fred; //declara uma referência a objeto
```

fred 

Objeto

O operador `new` aloca uma nova instância de objeto e retorna a referência (endereço) desse objeto na memória

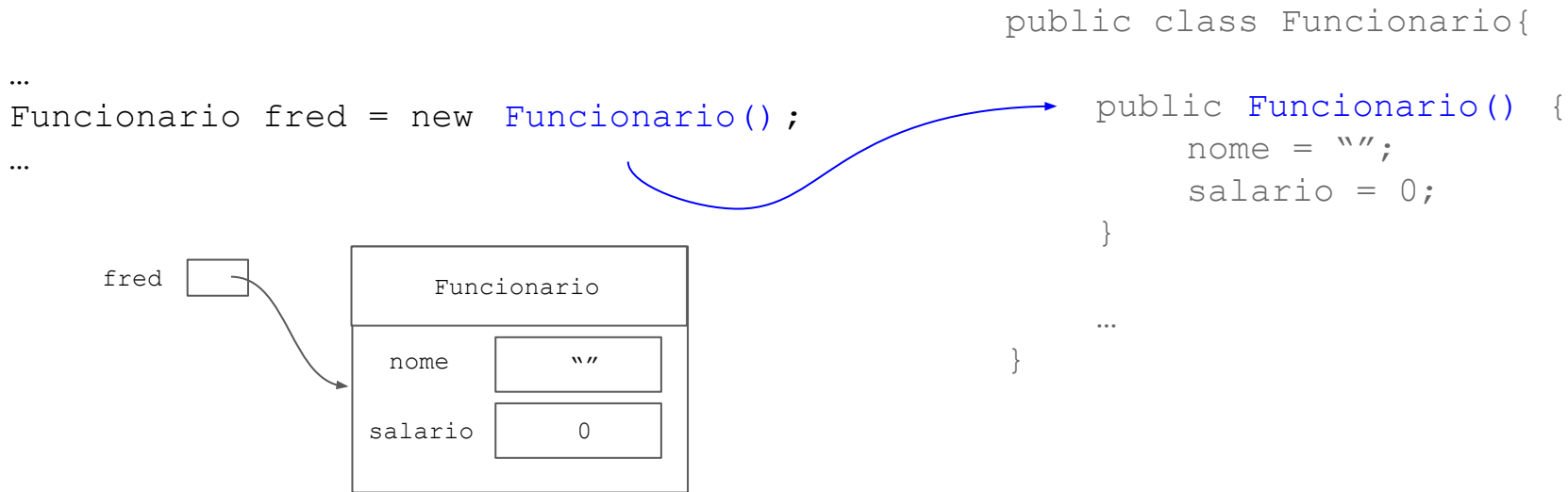
```
Funcionario fred; //declara uma referência a objeto  
fred = new Funcionario(); // aloca um objeto Funcionario
```



Inicialização

O **construtor** é o método que inicializa as variáveis de instância

É executado logo após a instanciação do objeto com o operador `new`



Inicialização

É possível **inicializar** as variáveis de instância **na declaração**

```
public class Funcionario {  
    public String nome = "";  
    ...  
}
```

Essa inicialização acontece depois que o objeto foi alocado e antes que o construtor execute

Inicialização

É possível ter **diferentes construtores** (sobrecarga de métodos)

```
public Funcionario(String nome, double salario)
{
    this.nome = nome;
    this.salario = salario;
}
public Funcionario() {
    nome = "";
    salario = 0;
}
```

O construtor que é chamado depende dos argumentos passados

Inicialização

Uma variável de instância declarada como `final` é inicializada e seu valor não pode ser alterado depois

```
public class Funcionario {  
    public final String nome;  
    ...  
}
```

Referência this

Quando um método é chamado a partir de um objeto, `this` recebe a referência deste objeto

```
public class Funcionario{  
  
    private String nome;  
    private double salario;  
  
    public Funcionario(String nome, double salario) {  
        this.nome = nome;  
        this.salario = salario;  
    }  
  
    ...  
}
```

Variável local com o mesmo nome de variável de instância

A referência `this` evita que a variável local “esconda” a variável de instância

Referência this

```
public class Funcionario{

    private String nome;
    private double salario;

    public void aumentaSalario(double porcentagem) {
        double aumento = this.salario * porcentagem / 100;
        this.salario += aumento;
    }

    ...

}
```


Bibliografia

Deitel, P.J. e Deitel, H.M. **Java How to Program: Late Objects**. 11th edition. Pearson, 2020.

C. S. Horstmann. **Core Java SE 9 for the Impatient**. Addison-Wesley, 2nd Edition, 2018.

Barnes, D. e Kolling, M. **Objects First with Java: A Practical Introduction Using BlueJ**. Pearson, 2016.