

Programação Orientada a Objetos

Andrés Menéndez

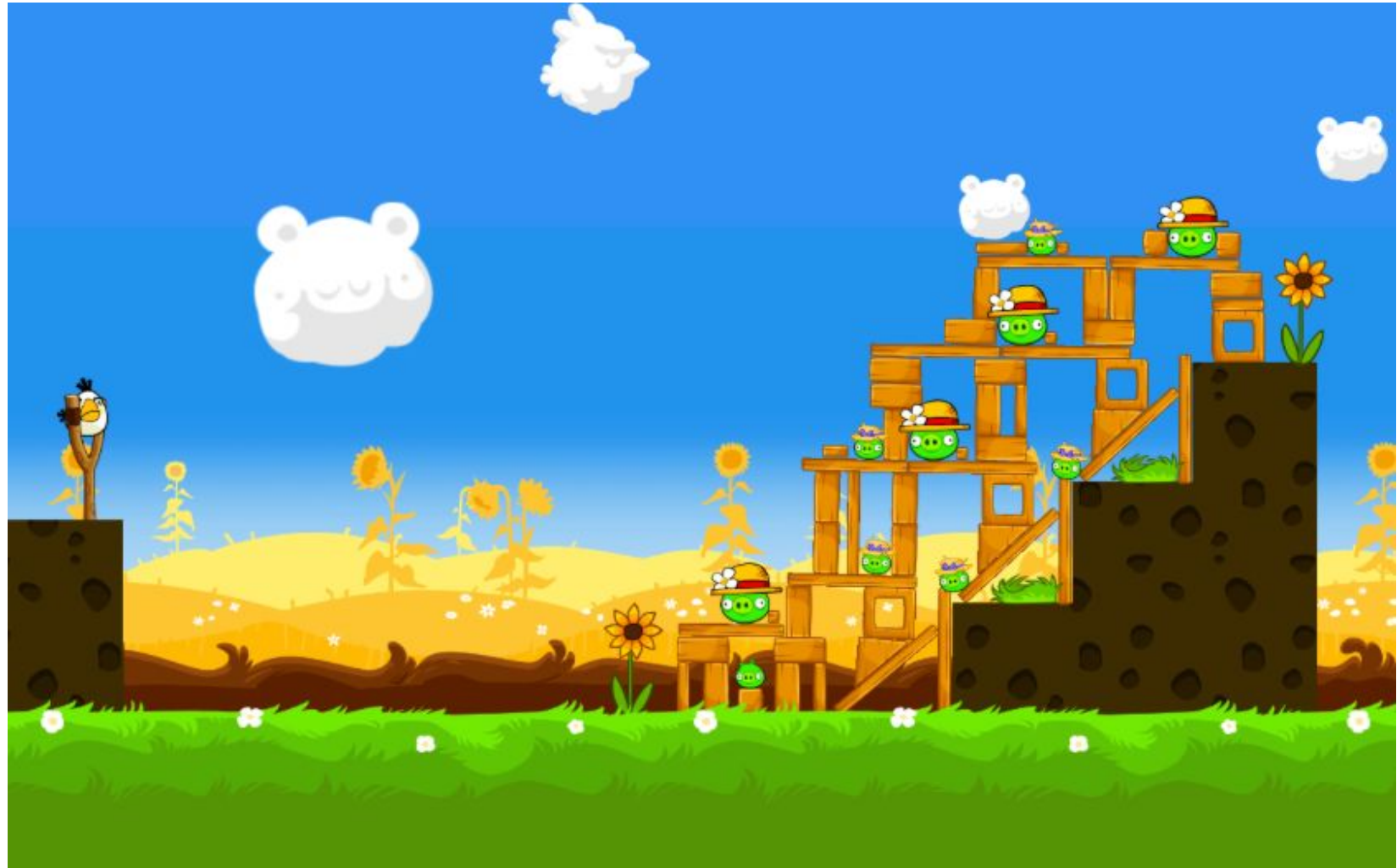
Departamento de Sistemas de Informação

Previously
 ***on***

Introdução

- Vamos falar sobre como interagir com os objetos que foram criados a partir das classes
- Para que os objetos tenham um determinado comportamento devemos chamar seus métodos
- Os métodos permitem que o objeto mude suas características (atributos)
- Vejamos nosso exemplo mais uma vez

Onde
usaremos
métodos?



Definição de métodos

- A definição de um método para uma classe Java segue a seguinte sintaxe:

```
<modificador> <tipo do retorno>  
    <nome do método> ([<lista de parâmetros>]) {  
    [<Implementação do método>]  
}
```

- Onde:
- Modificador: visibilidade do método, normalmente público ou privado
- Tipo de retorno: void ou algum tipo definido (int, boolean, String, etc.)
- Lista de parâmetros: conjunto de elementos “nome tipo” separado por vírgulas

Definição de métodos

- Suponha que temos uma classe relógio e vamos fazer o método para aumentar os segundos:

```
public void aumentarSegundos() {  
    segundos++;  
    if (segundos == 60)  
        segundos = 0;  
}
```

- Vamos analisar
 - O método é público (pode ser chamado externamente)
 - O método não definiu valor de retorno (void)
 - O método não tem parâmetros
 - O método modifica o atributo segundos

Definição de métodos

- Como seria um método para a classe Relógio se fosse acertar as horas?
- Uma possível implementação poderia ser

```
public void acertarHora(int hora, int minutos,  
                        int segundos) {  
    this.hora = hora;  
    this.minutos = minutos;  
    this.segundos = segundos;  
}
```

- Vamos analisar
 - Temos três parâmetros sendo passados
 - Usamos **this** para diferenciar parâmetros de atributos da classe



Definição de métodos

- O que acontece no método anterior se passarmos valores errados nos parâmetros? Por exemplo, se passarmos o valor 75 para os minutos...
- Se utilizarmos o método `acertarHora` da maneira que foi implementado teremos problemas com nosso objeto `Relógio`
- Sendo assim, o método precisa sofrer algumas modificações

Definição de métodos

- Vejamos

```
public boolean acertarHora(int hora, int min, int seg) {  
    if (hora > 23 || min > 59 || seg > 59) {  
        return false;  
    } else {  
        this.hora = hora;  
        this.min = min;  
        this.seg = seg;  
        return true;  
    }  
}
```

- Note que o método está retornando um boolean para indicar se os parâmetros passados estão corretos

Exercício

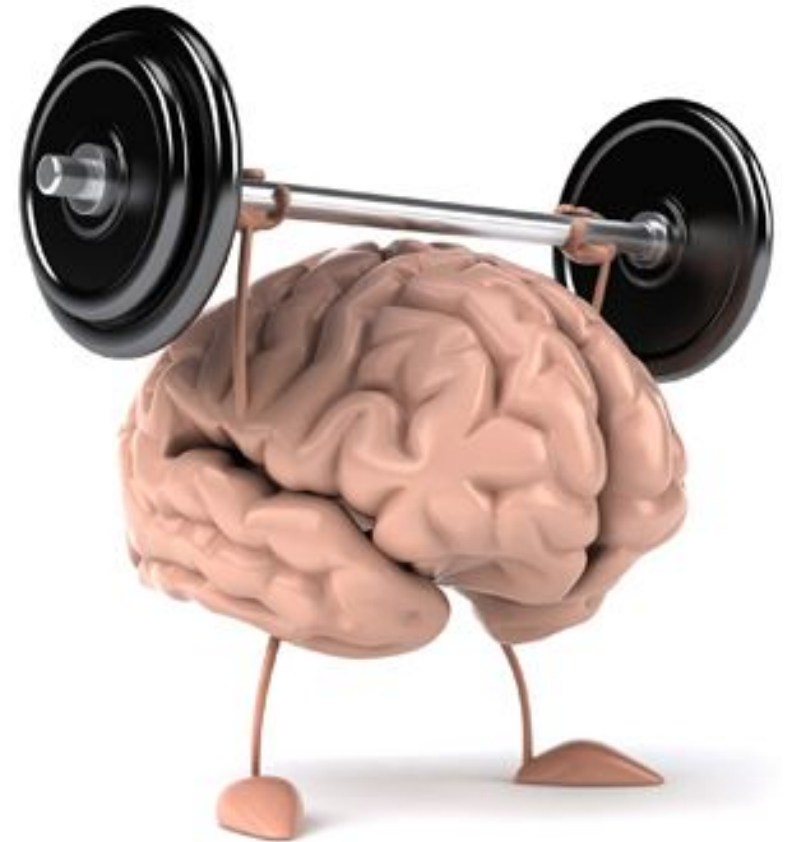
- Vamos fazer uma classe Relógio que permita guardar as horas, minutos e segundos
- A classe deverá ter os seguintes métodos:
 - Acertar o horário: permite definir uma determinada hora
 - Aumentar os segundos: aumenta os segundos, ajustando também os minutos e as horas
 - Mostrar hora: retorna uma string no formato HH:MM:SS

Exercício

- Faça um programa Java que crie um objeto do tipo relógio para testar o funcionamento do mesmo

Exercício++

- Faça um programa que crie um objeto do tipo Relógio e pergunte a hora atual
- O programa deve atualizar o relógio a cada segundo (use `Thread.sleep(1000)`) e mostrar as horas a cada 5 minutos



Métodos privados

São várias as situações onde podemos criar métodos que não ficarão visíveis externamente, são os chamados métodos privados

Estes métodos irão auxiliar métodos públicos na realização de tarefas

Vejamos novamente o método de aumentar os segundos do exemplo da classe Relógio

Métodos privados

- Se você fez o exercício, provavelmente modificou este método para ajustar também os minutos e as horas, aumentando a sua complexidade e a quantidade de linhas de código

```
public void aumentarSegundos() {  
    segundos++;  
}
```

- Utilizando métodos privados podemos deixar o método de aumentar os segundos com baixa complexidade

Métodos privados

- Vejamos como ficaria com métodos privados

```
public void aumentarSegundos() {  
    segundos++;  
    ajustarHorario();  
}
```

```
private void ajustarHorario() {  
    // lógica do método  
}
```

- O método ajustarHorário será encarregado de fazer as modificações necessárias nos atributos, deixando o método aumentarSegundos com uma baixa complexidade

A close-up photograph of a woman with long, dark, wavy hair and light blue eyes. She is resting her head on her right hand, with her index finger pointing upwards near her temple, suggesting a state of deep thought or contemplation. She is wearing a ring on her ring finger. The background is a plain, light-colored wall. The overall image has a soft, slightly desaturated aesthetic.

Vamos pensar um pouco

Overload de métodos

Em algumas situações podemos ter métodos com o mesmo nome, mas que tenham parâmetros diferentes

Os métodos com o mesmo nome são conhecidos como overload

Em português é chamado de sobrecarga, o que significa que aquele método poderá fazer coisas diferentes

Overload de métodos

- Vamos olhar um dos métodos do nosso exemplo do relógio

```
public boolean acertarHora(int hora, int min, int seg) {  
    if (hora > 23 || min > 59 || seg > 59) {  
        return false;  
    } else {  
        this.hora = hora;  
        this.min = min;  
        this.seg = seg;  
        return true;  
    }  
}
```

- Imagine que você deseja somente acertar a hora por causa do horário do verão

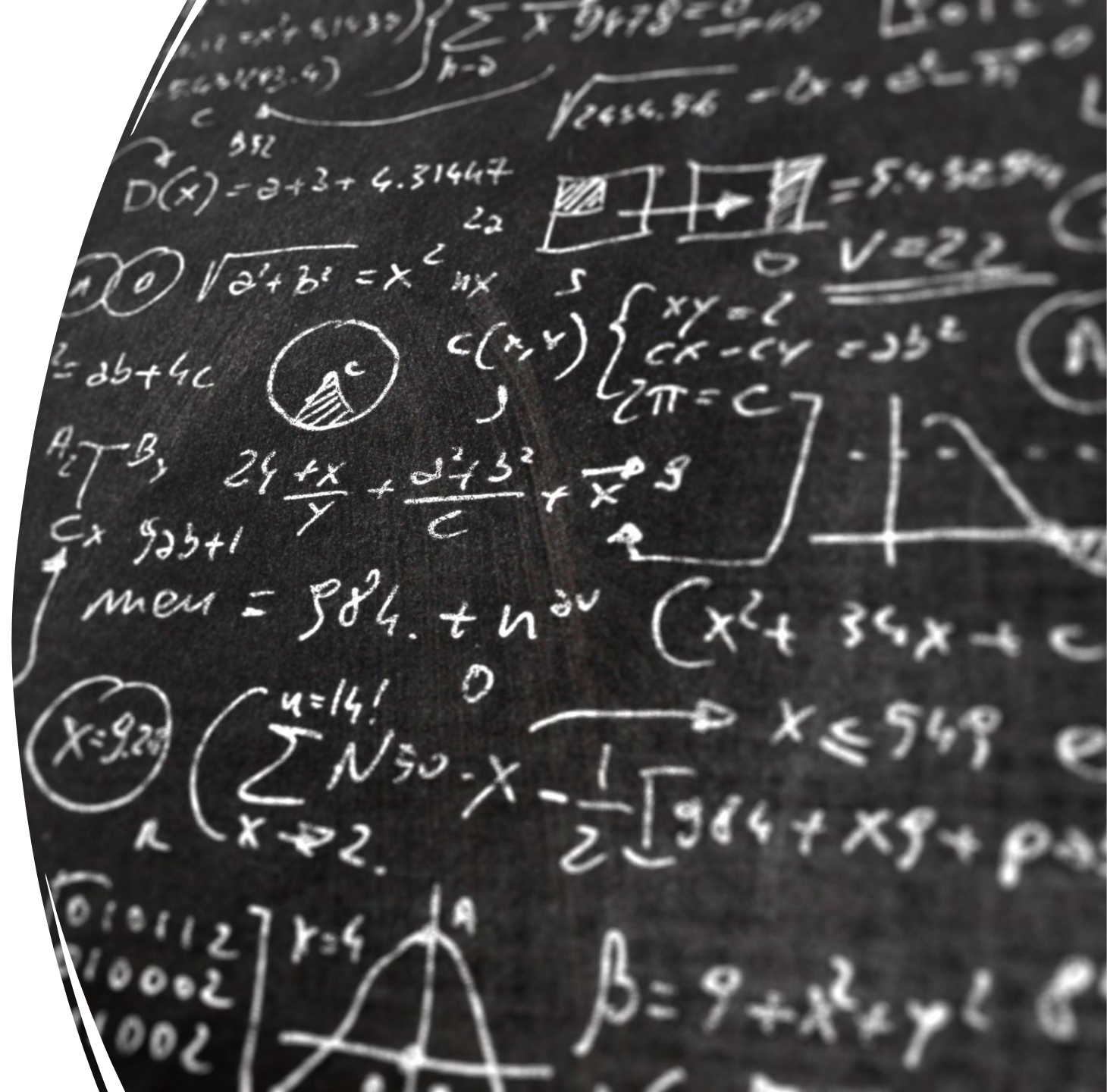
Overload de métodos

- Neste caso, bastaria a gente passar a nova hora, ao invés de ter que passar como parâmetro os minutos e os segundos. Sendo assim, nosso método poderia ficar assim:

```
public boolean acertarHora(int hora ) {  
    if (hora > 23) {  
        return false;  
    } else {  
        this.hora = hora;  
        return true;  
    }  
}
```

Overload de métodos

- A classe Relogio passará a ter dois métodos que possuem o mesmo nome, mas com implementação diferente



A close-up portrait of a young woman with long, dark, wavy hair and light blue eyes. She is resting her head on her right hand, with her index finger pointing upwards near her temple, suggesting a state of deep thought or contemplation. She is wearing a dark, patterned top. The background is a plain, light-colored wall. The overall mood is pensive and intellectual.

Vamos pensar um pouco

Exercício

- Faça uma classe Funcionario que permite armazenar a matricula, o nome e o salário base
- Os métodos da classe funcionário são:
 - Crie um construtor passando os parâmetros da classe
 - void atualizarSalario(double taxa) – permite aumentar o salário do funcionário. O parâmetro é o percentual de reajuste do salário.


Exercício

- Os métodos da classe funcionário são:
 - double calcularSalario() – retorna o salário que deverá ser pago
 - double calcularSalario(double descontos) – permite calcular o salário do funcionário, considerando os descontos que devem ser aplicados
 - double calcularSalario(double adicional, double descontos) - permite calcular o salário do funcionário, considerando um adicional e descontos que devem ser aplicados ao salário

cuidado!

SPOILERS



A close-up photograph of a Ferrari V8 engine, showing the red valve covers and silver intake manifolds. Below the engine, the red front grille of the car is visible, featuring the 'FERRARI' nameplate and the prancing horse emblem. An orange horizontal bar is located in the top left corner of the image.

Um motor é
um objeto
único?
