Programação Orientada a Objetos

Sobrecarga de Métodos

André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Abril 2020

Assinatura de um Método

- Formada por:
  - nome do método
  - número de parâmetros
  - tipos dos parâmetros (considerando a ordem)
- Métodos tenham o mesmo nome, mesmo número de parâmetros com os mesmos tipos (na ordem) têm a mesma assinatura

Sobrecarga de Métodos

- Técnica que envolve criar vários métodos com o **mesmo nome** e implementações diferentes

Tipos de Sobrecarga de Métodos

- Sobrecarga na mesma classe
  - assinaturas diferentes
- Sobrecarga em classes herdeiras
  - assinaturas podem ser iguais ou diferentes
  - tratado na aula de Polimorfismo

Tipos de Sobrecarga de Métodos
Esta Aula

- **Sobrecarga na mesma classe**
  - **assinaturas diferentes**
- Sobrecarga em classes herdeiras
  - assinaturas podem ser iguais ou diferentes
  - tratado na aula de Polimorfismo

Compare as Assinaturas

```
String metodo()
```

```
int metodo(int a, int b)
```

```
int metodo(int a, int b, String c)
```

```
int metodo(int a, String c, int b)
```

Assinaturas Diferentes

```
String metodo()
```

⇳ retornos diferentes; parâmetros diferentes

```
int metodo(int a, int b)
```

⇳ parâmetros diferentes

```
int metodo(int a, int b, String c)
```

⇳ parâmetros diferentes (ordem dos tipos)

```
int metodo(int a, String c, int b)
```

Diferenciação na Chamada

```
String metodo()
int metodo(int a, int b)
int metodo(int a, int b, String c)
int metodo(int a, String c, int b)
```

Chamada:

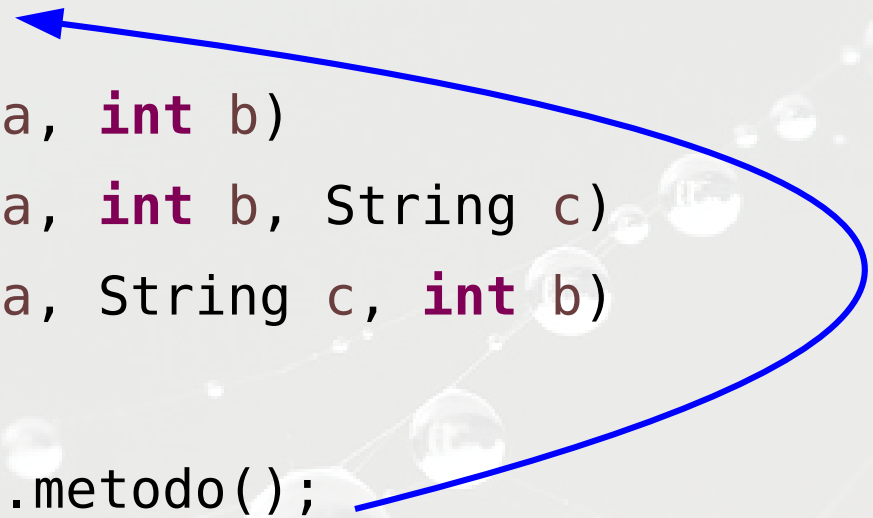
```
String r1 = app.metodo();
int r2 = app.metodo(10, 15);
int r3 = app.metodo(10, 15, "texto");
int r4 = app.metodo(10, "texto", 15);
```

Diferenciação na Chamada

```
String metodo()
int metodo(int a, int b)
int metodo(int a, int b, String c)
int metodo(int a, String c, int b)
```

Chamada:

```
String r1 = app.metodo();
int r2 = app.metodo(10, 15);
int r3 = app.metodo(10, 15, "texto");
int r4 = app.metodo(10, "texto", 15);
```

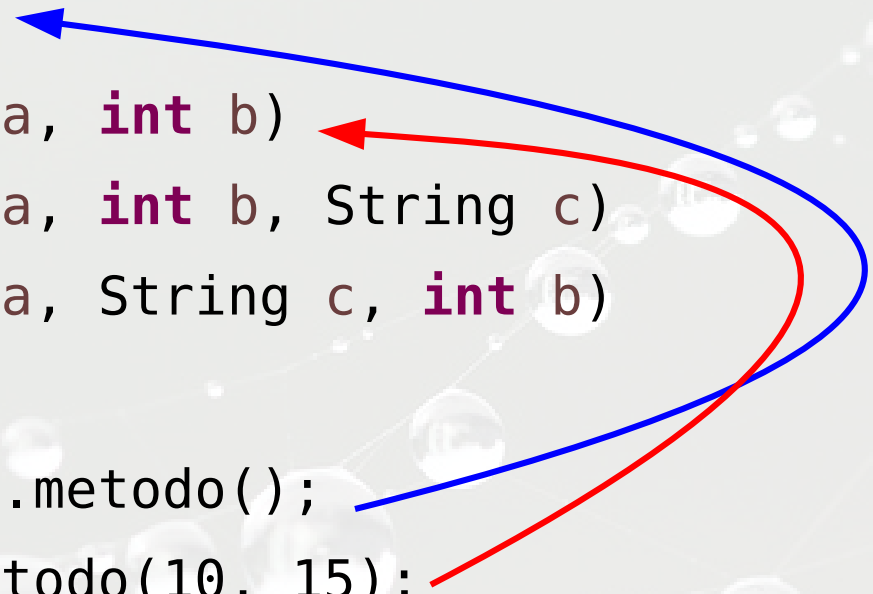


Diferenciação na Chamada

```
String metodo()
int metodo(int a, int b)
int metodo(int a, int b, String c)
int metodo(int a, String c, int b)
```

Chamada:

```
String r1 = app.metodo();
int r2 = app.metodo(10, 15);
int r3 = app.metodo(10, 15, "texto");
int r4 = app.metodo(10, "texto", 15);
```

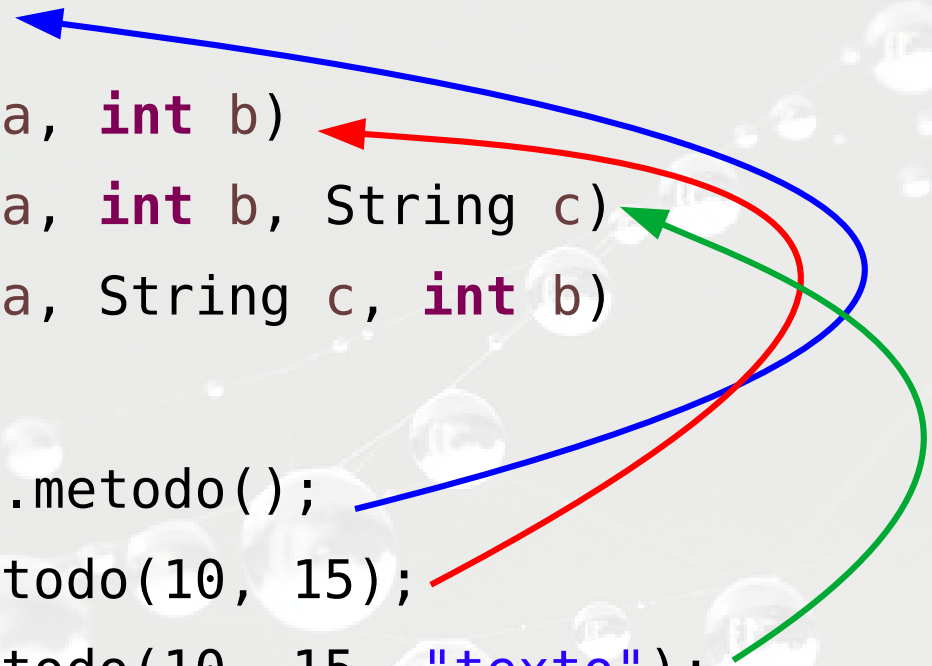


Diferenciação na Chamada

```
String metodo()
int metodo(int a, int b)
int metodo(int a, int b, String c)
int metodo(int a, String c, int b)
```

Chamada:

```
String r1 = app.metodo();
int r2 = app.metodo(10, 15);
int r3 = app.metodo(10, 15, "texto");
int r4 = app.metodo(10, "texto", 15);
```

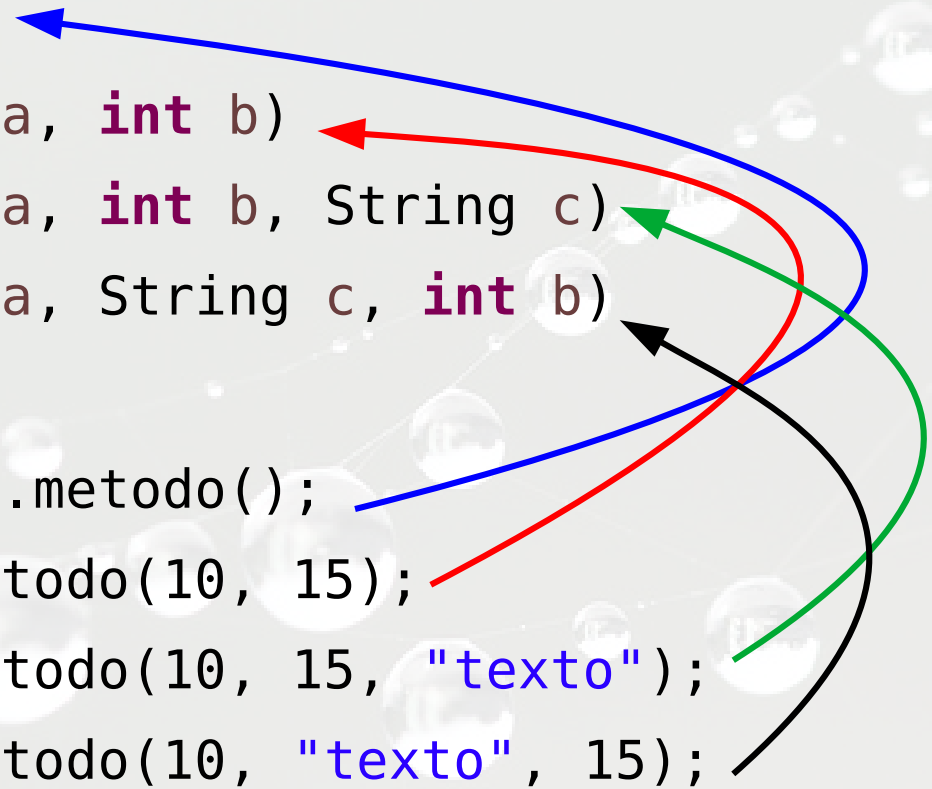


Diferenciação na Chamada

```
String metodo()
int metodo(int a, int b)
int metodo(int a, int b, String c)
int metodo(int a, String c, int b)
```

Chamada:

```
String r1 = app.metodo();
int r2 = app.metodo(10, 15);
int r3 = app.metodo(10, 15, "texto");
int r4 = app.metodo(10, "texto", 15);
```



Compare as Assinaturas

```
int metodo(int a, int b, String c)

int metodo(int x, int y, String z)
```

Assinaturas Iguais

```
int metodo(int a, int b, String c)
```

⇳ assinaturas iguais (o nome do parâmetro não importa)

```
int metodo(int x, int y, String z)
```

Não Diferenciável na Chamada

```
int metodo(int a, int b, String c)

int metodo(int x, int y, String z)
```

Chamada:

```
int r3 = app.metodo(10, 15, "texto");
```

Não Diferenciável na Chamada

```
int metodo(int a, int b, String c)

int metodo(int x, int y, String z)

Chamada:

int r3 = app.metodo(10, 15, "texto");
```

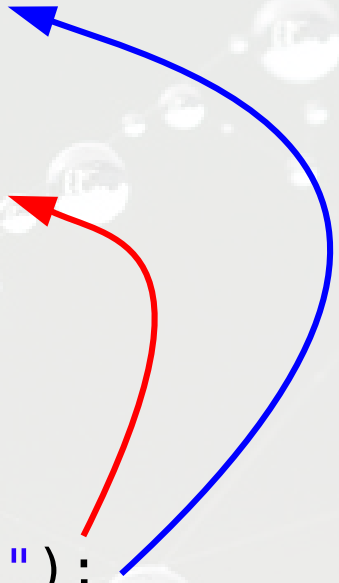
Não Diferenciável na Chamada

```
int metodo(int a, int b, String c)

int metodo(int x, int y, String z)

Chamada:

int r3 = app.metodo(10, 15, "texto");
```



Compare as Assinaturas

```
int metodo(int a, int b, String c)

String metodo(int a, int b, String c)
```

Assinaturas Inadequadas para Sobrecarga

```
int metodo(int a, int b, String c)
```

⇳ assinaturas diferentes por causa do tipo de retorno

```
String metodo(int a, int b, String c)
```

- Esta diferença entre as assinaturas não pode ser adotada em sobrecarga, porque não há como diferenciá-las na chamada

Assinaturas Inadequadas para Sobrecarga

```
int metodo(int a, int b, String c)

String metodo(int a, int b, String c)
```

Chamada:

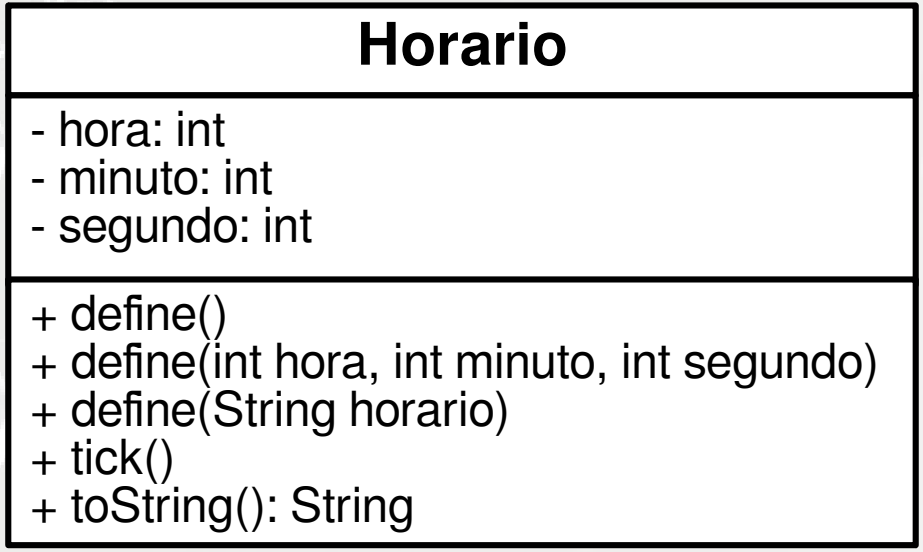
```
... app.metodo(10, 15, "texto");
```

- Tipo de retorno não é usado na verificação da chamada

Para que serve Sobrecarga na Mesma Classe?

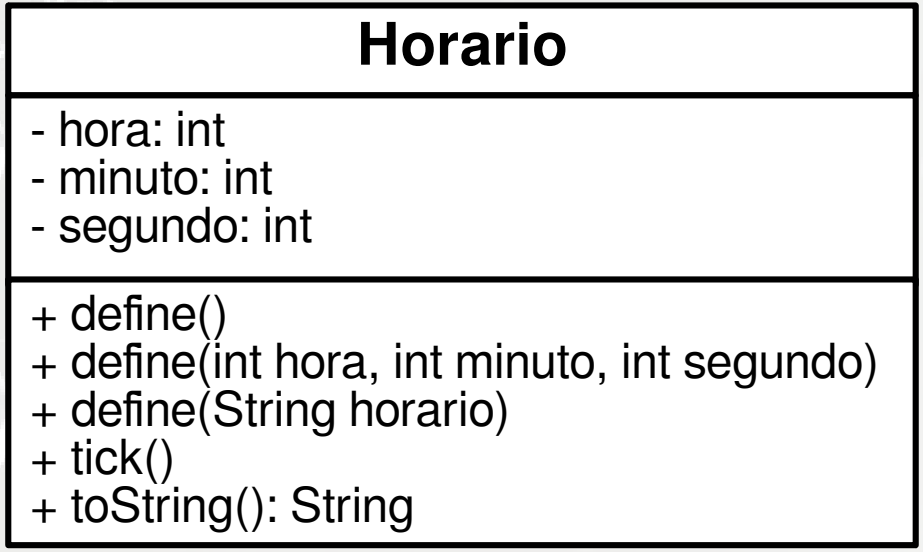
- Mecanismo de publicar variantes do mesmo serviço
  - mesmo nome de método = mesmo serviço
  - recomendação não controlada pela linguagem
- Diferentes assinaturas do método com o mesmo nome
  - formas diferentes de se requisitar o mesmo serviço
  - pode indicar variantes sobre o mesmo serviço

Sobrecarga em Horário



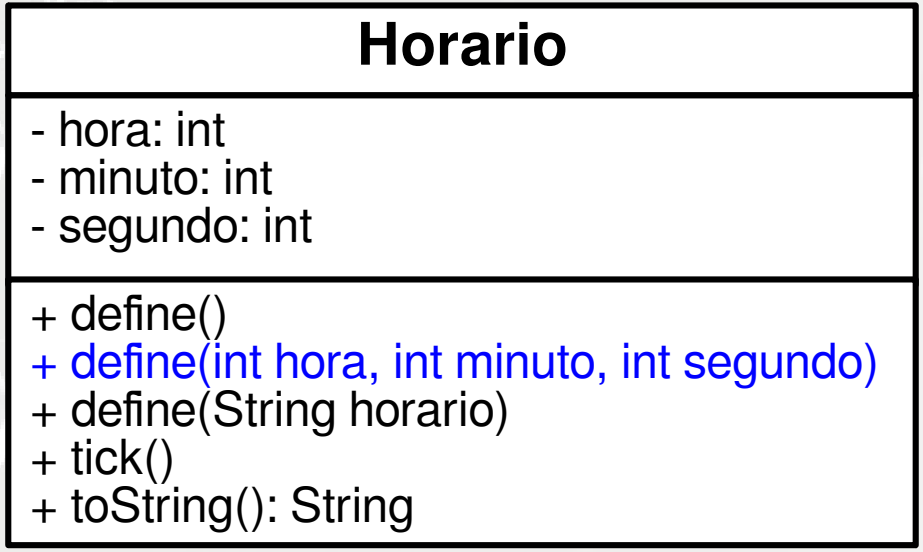
Sobrecarga em Horário
Versão A

- Cada método define atualiza os atributos por sua conta



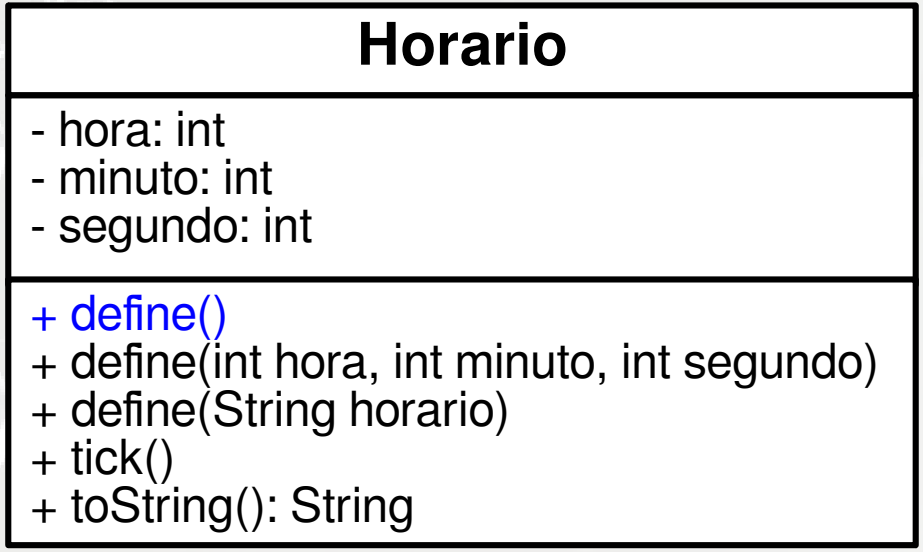
Sobrecarga em Horário

```
public class Horario {
    private int hora, minuto, segundo;
    ...
    public void define(int hora, int minuto, int segundo) {
        this.hora = hora;
        this.minuto = minuto;
        this.segundo = segundo;
    }
```



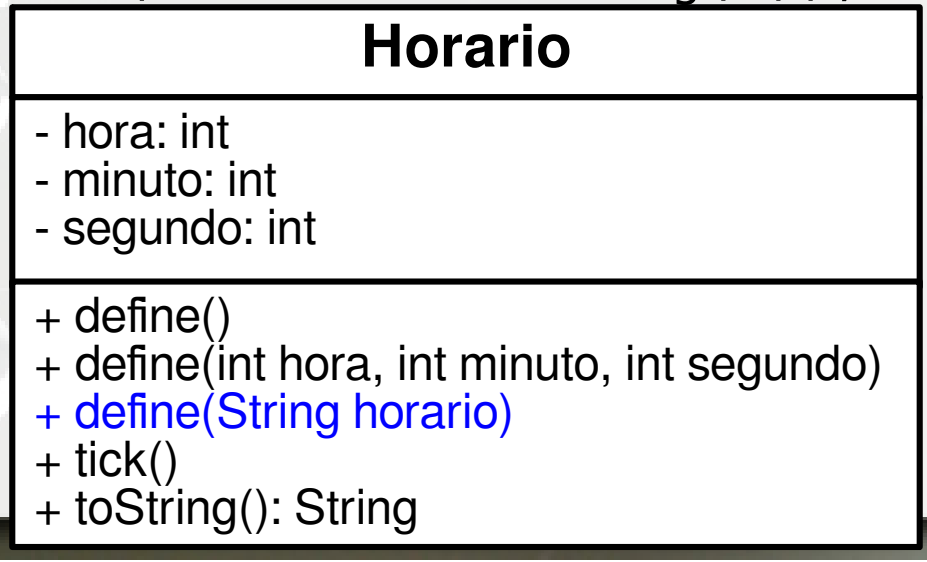
Auto-referência em Sobrecarga

```
public class Horario {
    ...
    public void define() {
        this.hora = 0;
        this.minuto = 0;
        this.segundo = 0;
    }
```



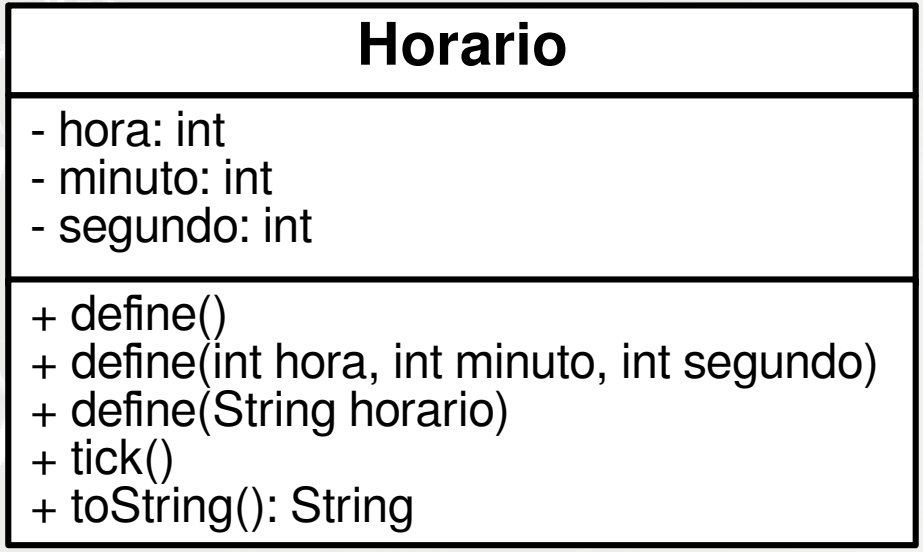
Sobrecarga em Horário

```
...
public void define(String horario) {
    this.hora = Integer.parseInt(horario.substring(0,2));
    this.minuto = Integer.parseInt(horario.substring(3,5));
    this.segundo = Integer.parseInt(horario.substring(6));
}
```



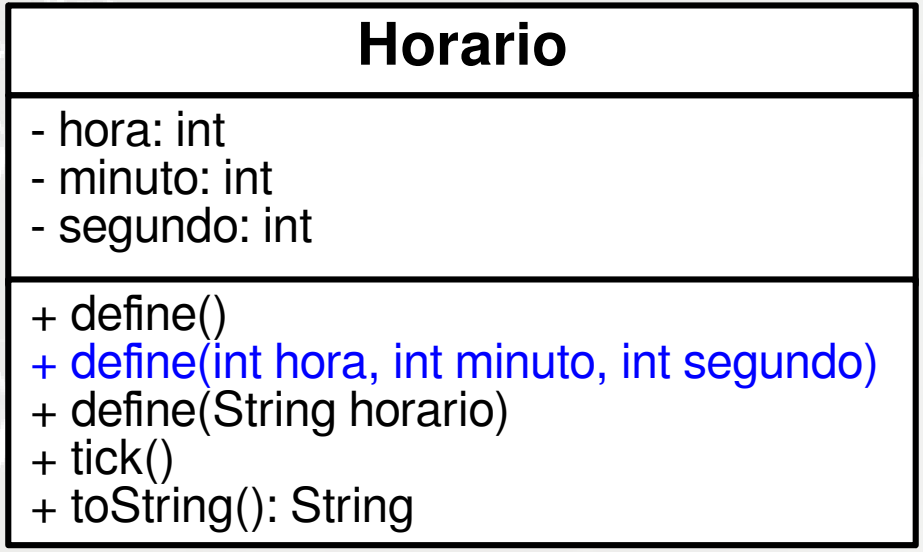
Sobrecarga em Horário
Versão B

- Métodos sobrecarregados concentram a atualização dos atributos em um deles



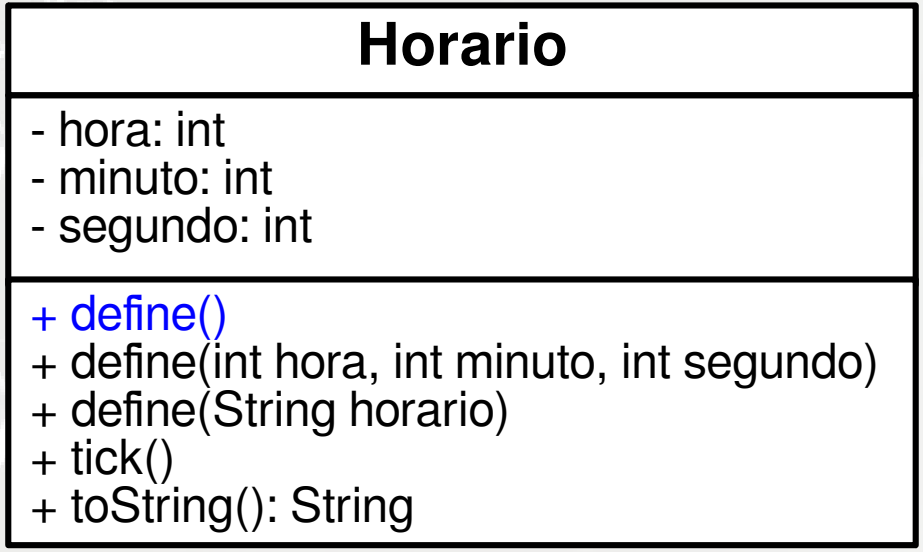
Sobrecarga em Horário

```
public class Horario {
    private int hora, minuto, segundo;
    ...
    public void define(int hora, int minuto, int segundo) {
        this.hora = hora;
        this.minuto = minuto;
        this.segundo = segundo;
    }
```



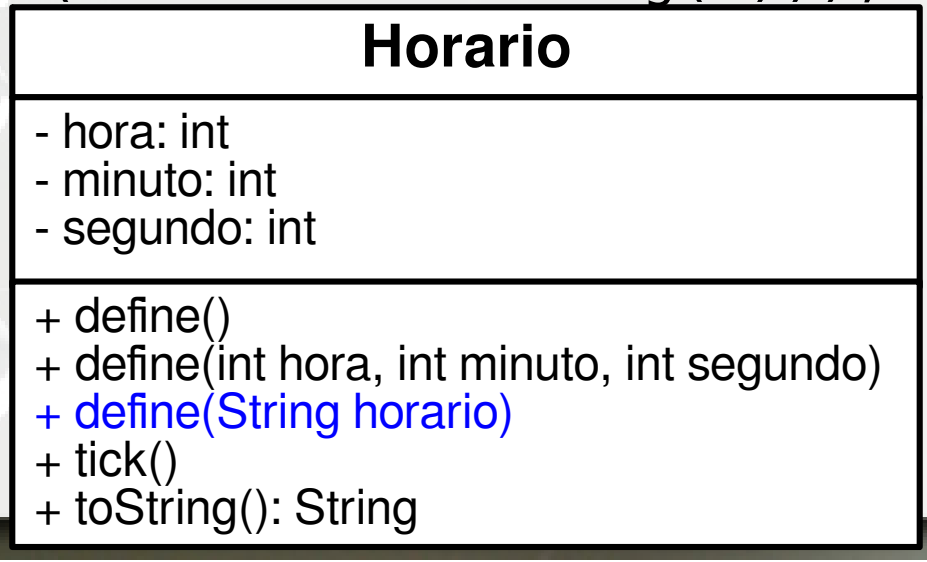
Auto-referência em Sobrecarga

```
public class Horario {
    ...
    public void define() {
        this.define(0, 0, 0);
    }
```



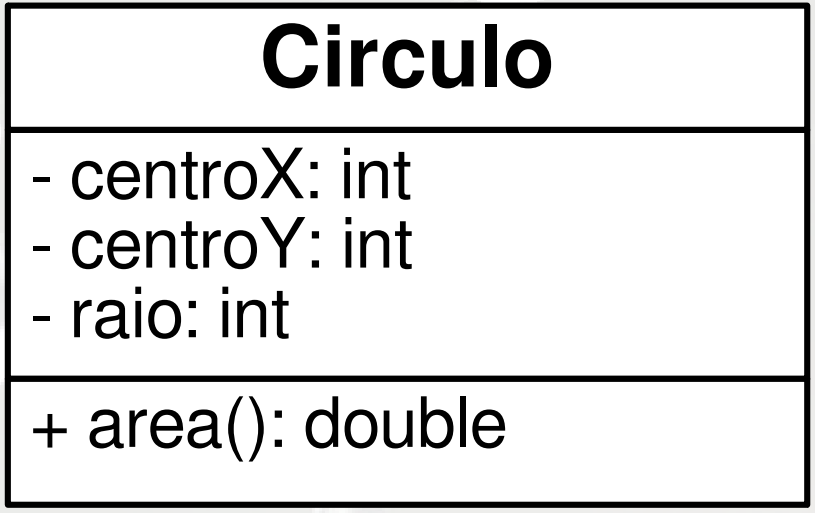
Sobrecarga em Horário

```
...
public void define(String horario) {
 this.define(Integer.parseInt(horario.substring(0,2)),
             Integer.parseInt(horario.substring(3,5)),
             Integer.parseInt(horario.substring(6)));
}
```



Tarefa `Circulo`
Sobrecarga de Método

- Dada a classe `Circulo` apresentada anteriormente (versão sem construtor), escreva dois métodos `define` sobrecarregados que inicializem `centroX`, `centroY` e `raio`

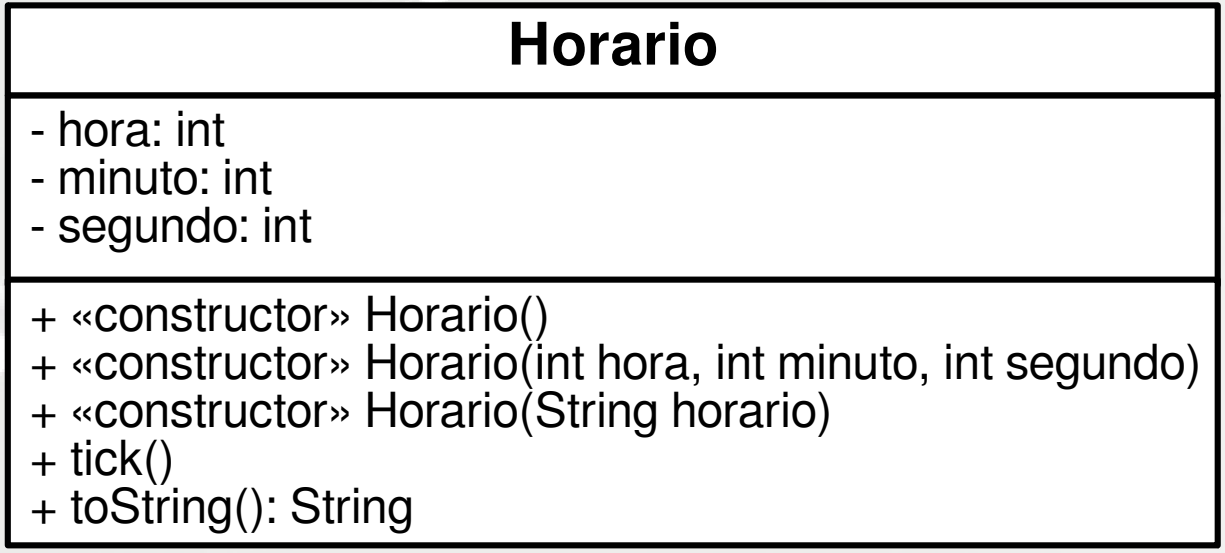


Sobrecarga de Construtor

- Construtor também pode ser sobrecarregado:

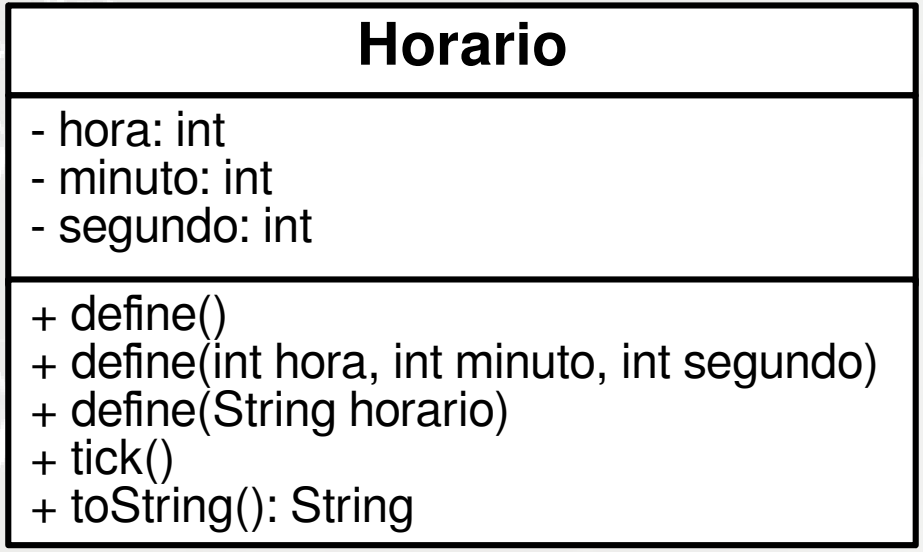
```
public class AppSobrecarga01 {
    AppSobrecarga01() {…}
    AppSobrecarga01(int a, int b) {…}
    AppSobrecarga01(int a, int b, String c) {…}
    AppSobrecarga01(int a, String c, int b) {…}
```

Sobrecarga do Construtor em Horário



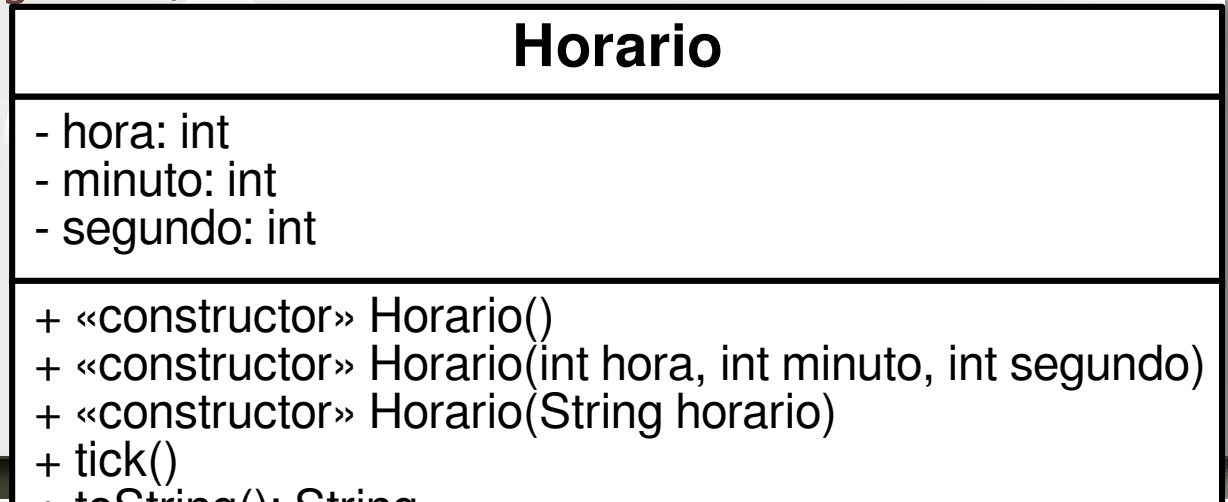
Sobrecarga do Construtor em Horário
Versão A

- Cada construtor inicializa os atributos por sua conta



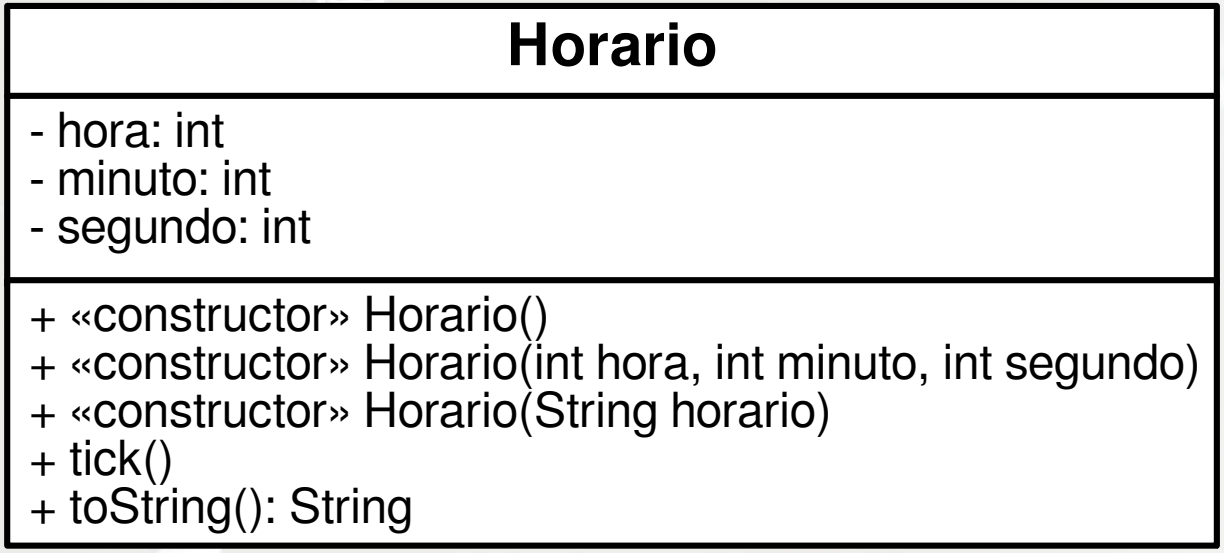
Sobrecarga do Construtor em Horário

```
public class Horario {
    private int hora, minuto, segundo;
    public Horario(int hora, int minuto, int segundo) {
        this.hora = hora;
        this.minuto = minuto;
        this.segundo = segundo;
    }
```



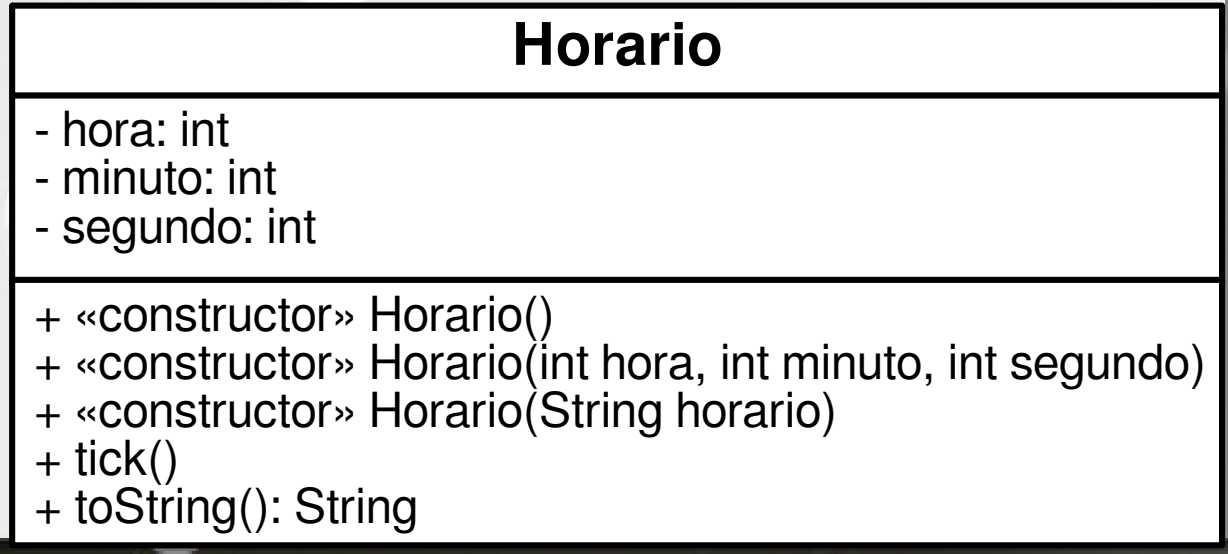
Sobrecarga do Construtor em Horário

```
...
public Horario() {
    this.hora = 0;
    this.minuto = 0;
    this.segundo = 0;
}
```



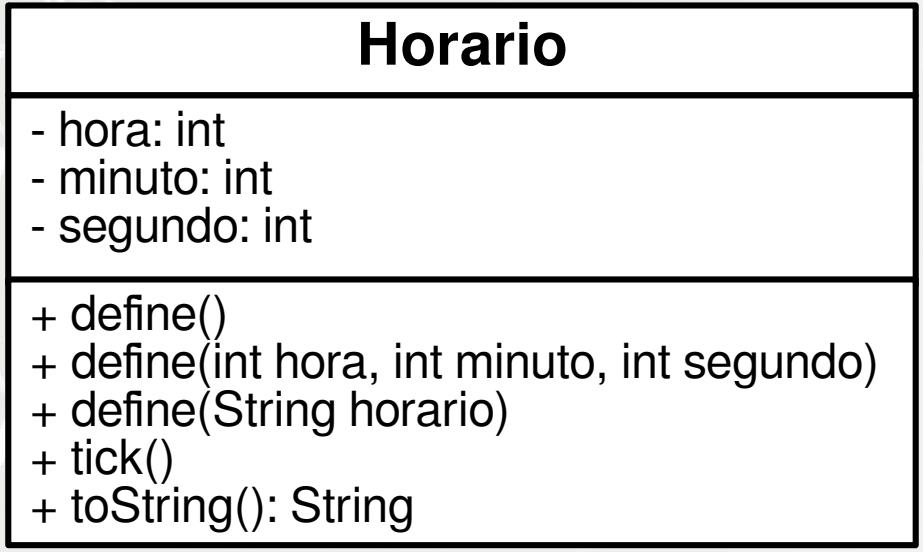
Sobrecarga do Construtor em Horário

```
...
public Horario(String horario) {
    this.hora = Integer.parseInt(horario.substring(0,2));
    this.minuto = Integer.parseInt(horario.substring(3,5));
    this.segundo = Integer.parseInt(horario.substring(6));
}
```



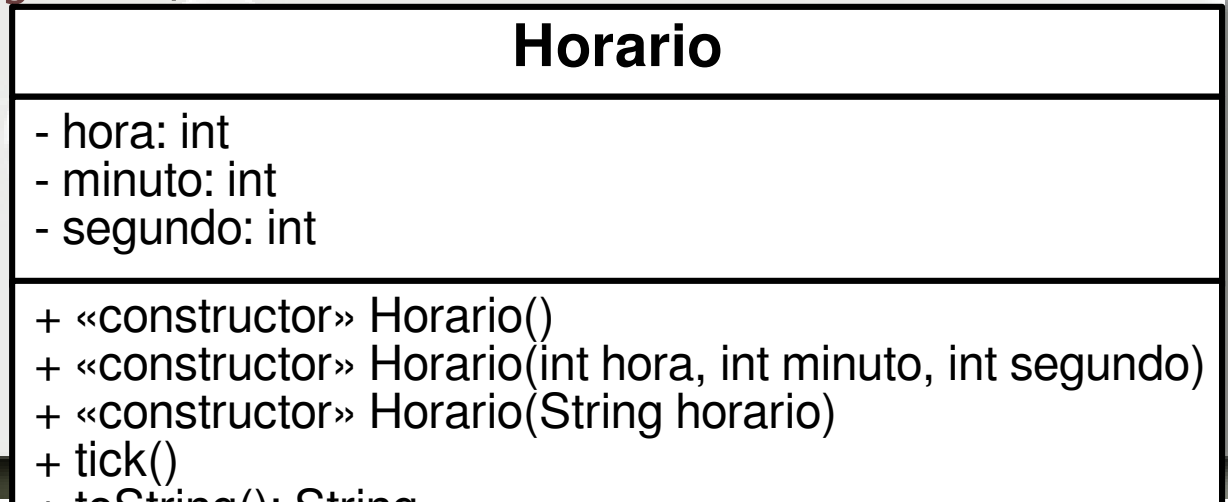
Sobrecarga do Construtor em Horário
Versão B

- Construtores sobrecarregados concentram a atribuição em um deles



Sobrecarga do Construtor em Horário

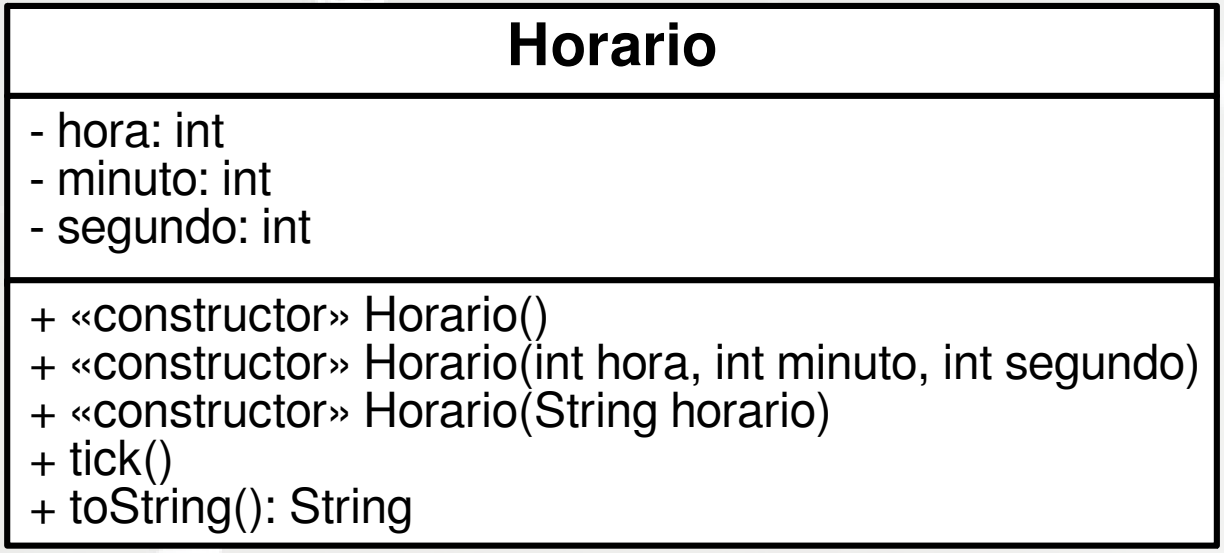
```
public class Horario {
    private int hora, minuto, segundo;
    public Horario(int hora, int minuto, int segundo) {
        this.hora = hora;
        this.minuto = minuto;
        this.segundo = segundo;
    }
```



Auto-referência do Construtor na Mesma Classe

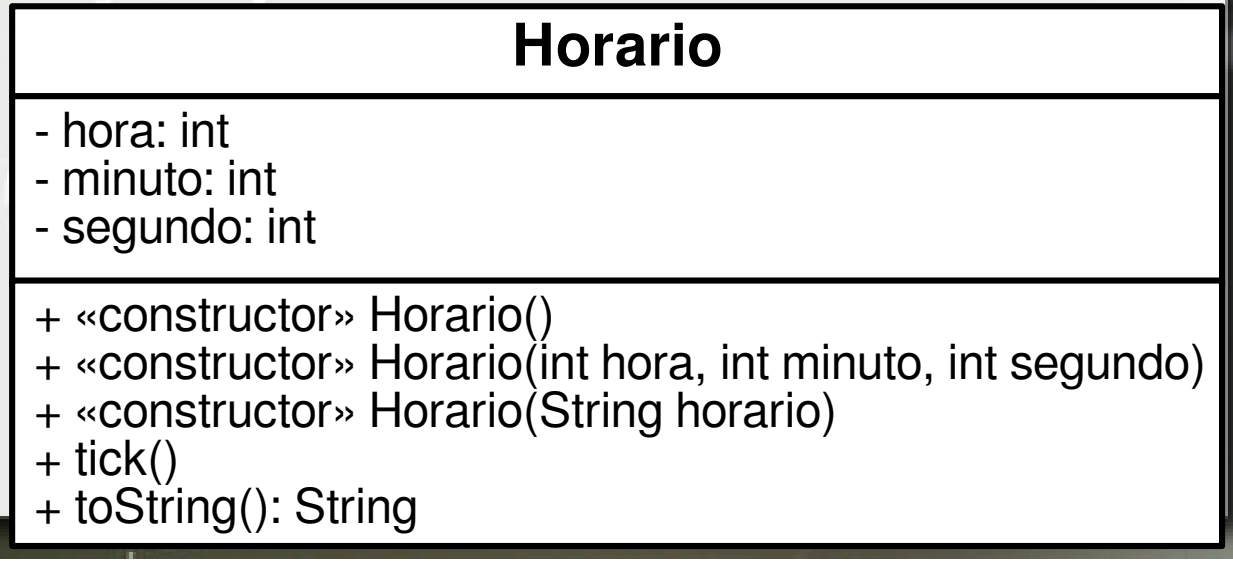
`this`

```
...
public Horario() {
    this(0, 0, 0);
}
```



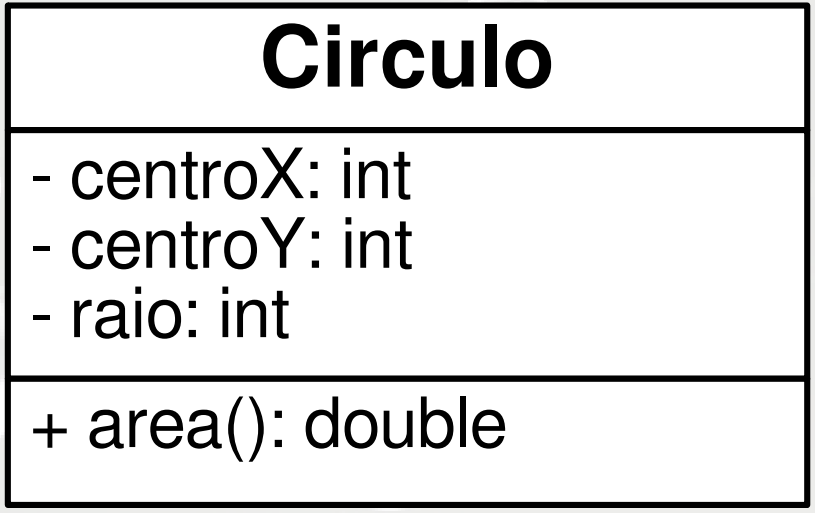
Sobrecarga do Construtor em Horário

```
...
public Horario(String horario) {
    this(Integer.parseInt(horario.substring(0,2)),
         Integer.parseInt(horario.substring(3,5)),
         Integer.parseInt(horario.substring(6)));
}
```



Tarefa `Circulo`
Sobrecarga de Construtor

- Adapte a tarefa anterior de modo que a inicialização de `centroX`, `centroY` e `raio` possa ser feita por dois construtores sobrecarregados



Tipos de Sobrecarga de Métodos
Próxima Aula - Polimorfismo

- Sobrecarga na mesma classe
  - assinaturas diferentes
- **Sobrecarga em classes herdeiras**
  - **assinaturas podem ser iguais ou diferentes**
  - **tratado na aula de Polimorfismo**

André Santanchè
http://www.ic.unicamp.br/~santanche

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: http://creativecommons.org/licenses/by-nc-sa/3.0/
- Agradecimento a James Ratcliffe [http://www.flickr.com/photos/jamie/1762955591/] por sua fotografia “A spider web after a misty morning” usada na capa e nos fundos, disponível em [http://www.flickr.com/photos/jamie/1762955591/]

  vide licença específica da fotografia.