Orientação a Objetos

Classes, Objetos, Atributos e Encapsulamento

Por qual motivo devemos utilizar a Orientação a Objetos?

Motivação

- Um sistema acadêmico deve controlar, para cada semestre, as turmas abertas e os respectivos alunos matriculados em cada turma. Vamos representar essas informações de forma estruturada, assim como fazíamos na disciplina Algoritmos?

Introdução às Classes

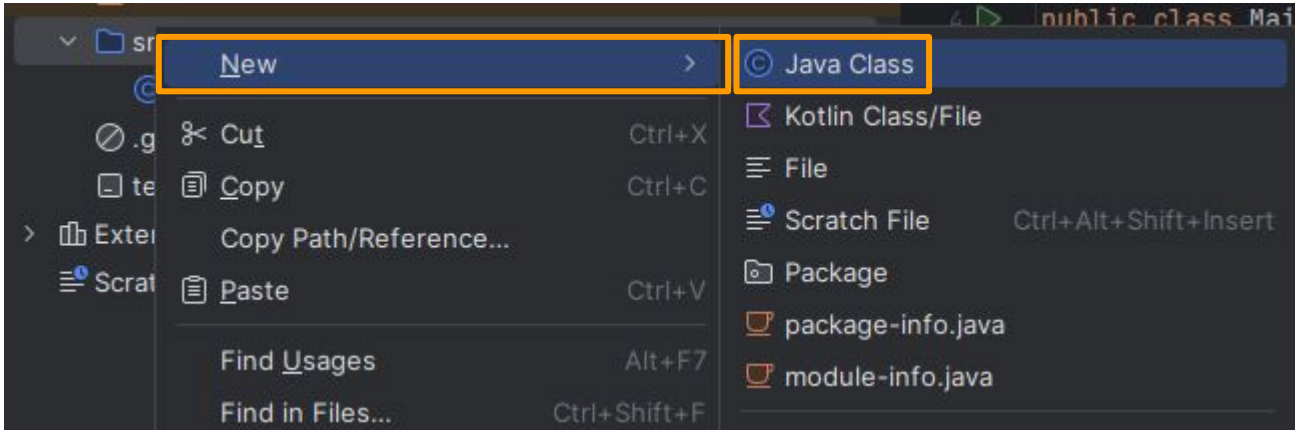
Criando a Primeira Classes na IDE

Classes

- Na programação Orientada a Objetos, é possível definir um tipo através de uma Classe.
- Uma classe define atributos e operações.
  - Os atributos controlam o estado da classe.
  - As operações são ações que podem ser efetuadas pelos objetos instanciados.

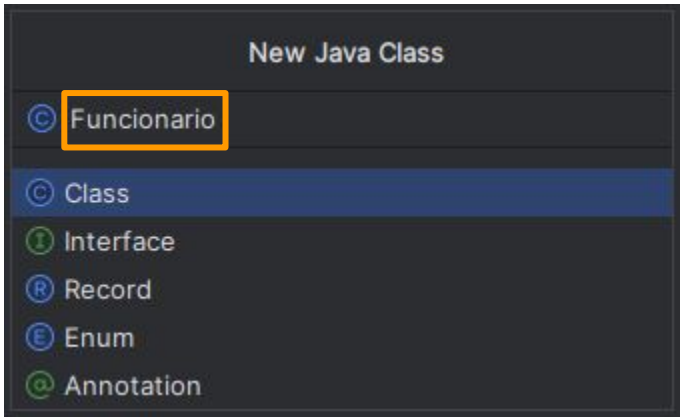
Classes

- Para criar uma classe no IntelliJ, clique com o botão direito do mouse em cima da pasta src e selecione:



Classes

- Digite o nome “Funcionario” para a classe e tecle <Enter>



Atenção: evite acentos e outros caracteres

Classes

- Veja o código criado:

```
public class Funcionario {

}
```

Classes

- Veja o código criado:

```
public class Funcionario {

}
```

Uma classe é declarada com a palavra reservada class.

Classes

- Veja o código criado:

```
public class Funcionario {

}
```

- Nome ou identificador da classe.
  - Deve ser utilizado o UpperCamelCase, ou seja:
    - O nome da classe deve ser iniciado por uma letra maiúscula.
    - A cada nova palavra, a inicial da mesma deve ser maiúscula.

Classes

- Veja o código criado:

```
public class FuncionarioMensalista {

}
```

- Nome ou identificador da classe.
  - Deve ser utilizado o UpperCamelCase, ou seja:
    - O nome da classe deve ser iniciado por uma letra maiúscula.
    - A cada nova palavra, a inicial da mesma deve ser maiúscula.

Classes

- Veja o código criado:

```
public class Funcionario {

}
```

- Modificador de acesso ou modificador de visibilidade
  - Quando public, indica que esta classe é pública e pode ser acessada por outras classes e objetos em seu sistema.

Classes

- Veja o código criado:

```
public class Funcionario {

}
```

- Delimitadores de escopo.
  - Os atributos e operações existentes na classe devem estar entre { e }.
  - Note que cada operação existente na classe também pode ter seus próprios delimitadores de acesso (chaves).

Atributos

Controlando o Estado de um Objeto

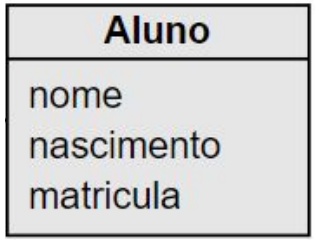
Atributos

- Atributos são como se fossem variáveis pertencentes à um objeto.

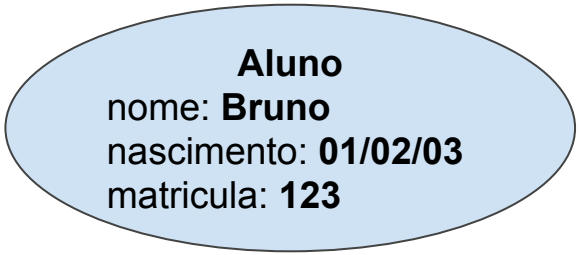
- Ao codificar um atributo em uma classe, cada instância dessa classe (cada objeto instanciado a partir dessa classe) possuirá este atributo e cada objeto manterá os valores de seus próprios atributos.

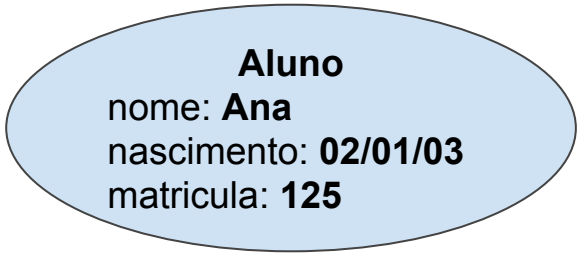
Atributos

- Exemplo de classe com atributos:



- Exemplos de objetos instanciados a partir desta classe:





Atributos

- Em nossa classe, criaremos os seguintes atributos:

```
public class Funcionario {
    int codigo;
    String nome;
}
```

Instanciando Objetos

- Um objeto é uma instância de uma classe.
  - Ele é criado em acordo com a especificação constante na classe.
  - Veja à seguir como podemos instanciar um objeto do tipo funcionário no main.

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo = 1;
        func1.nome = "Bruno";

        System.out.printf("%d - %s",
                          func1.codigo, func1.nome);
    }
}
```

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo
    }
}
```

- **func1** é uma variável do tipo **Funcionario**.
  - Imagine que você deseja declarar uma variável para armazenar um inteiro que é a idade de uma pessoa. A declaração seria assim:
    int idade;
    - int é o tipo
    - idade é o nome ou identificador da variável

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo
    }
}
```

- **func1** é uma variável do tipo **Funcionario**.
  - Agora uma variável para armazenar um funcionário, poderia ser declarado assim:
    Funcionario objeto;
    - **Funcionario** é o tipo
    - objeto é o nome ou identificador da variável

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo = 1;
        fun

        Sys

    }
}
```

- O operador **new** instancia um objeto.
  - Após este operador, deve ser chamado o construtor de uma classe.
    - O assunto construtores será visto mais à frente.
      - Por enquanto considere que à frente do operador **new** deve constar o nome da classe que se deseja instanciar seguido de ().

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo = 1;
        func1.nome = "Bruno";
```

- Sempre que você utilizar o nome da variável declarada, você estará referenciando o objeto em questão.
- Para acessar um atributo deste objeto, basta utilizar o ponto (.) após a referência do mesmo.
- Na linha destacada, o atributo **codigo** está sendo utilizado. Como o mesmo se encontra à esquerda do operador de atribuição, o valor 1 está sendo atribuído ao atributo **codigo** do funcionário em questão.

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo = 1;
        func1.nome = "Bruno";


                                                func1.nome);
    }
}
```

- O nome do funcionário em questão foi alterado para Bruno.

Instanciando Objetos

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.codigo = 1;
        func1.nome = "Bruno";

        System.out.printf("%d - %s",
                          func1.codigo, func1.nome);
    }
}
```

- Leitura dos atributos **codigo** e **nome** do funcionário em questão.
- Lembre-se: à esquerda do operador de atribuição, está atribuindo um valor. Em outras posições, está lendo o valor.

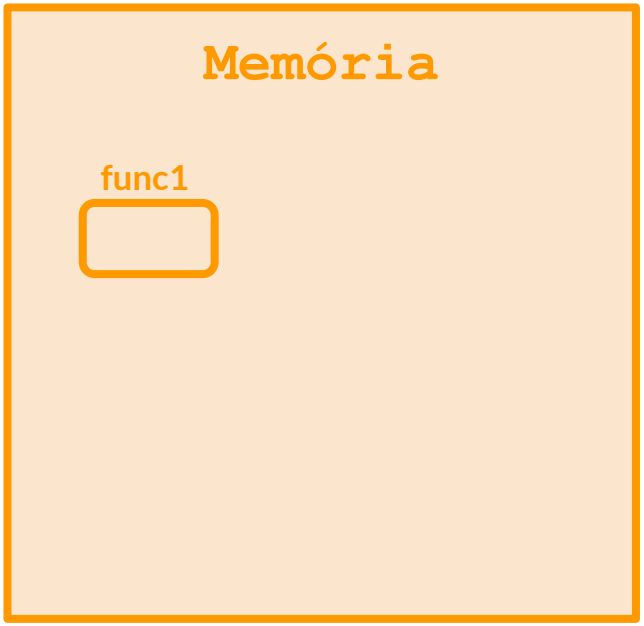
Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```

Memória

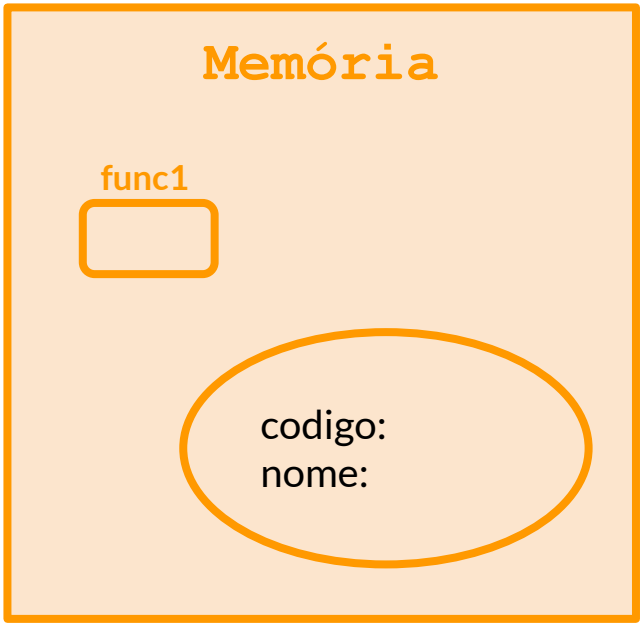
Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```



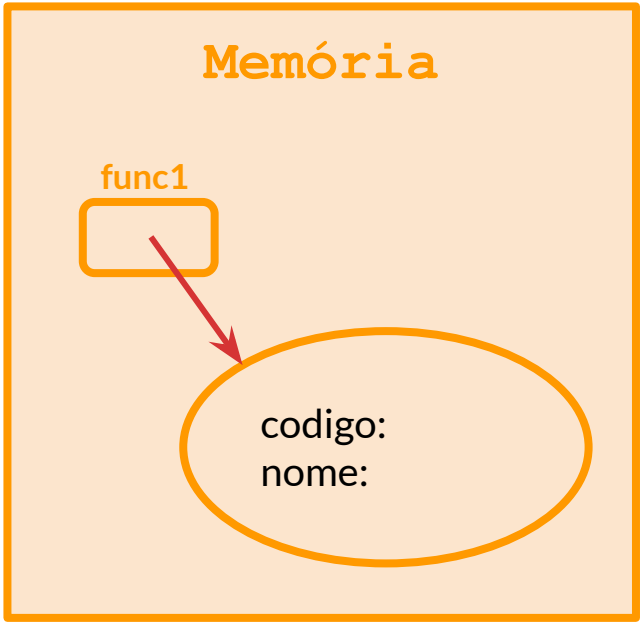
Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```



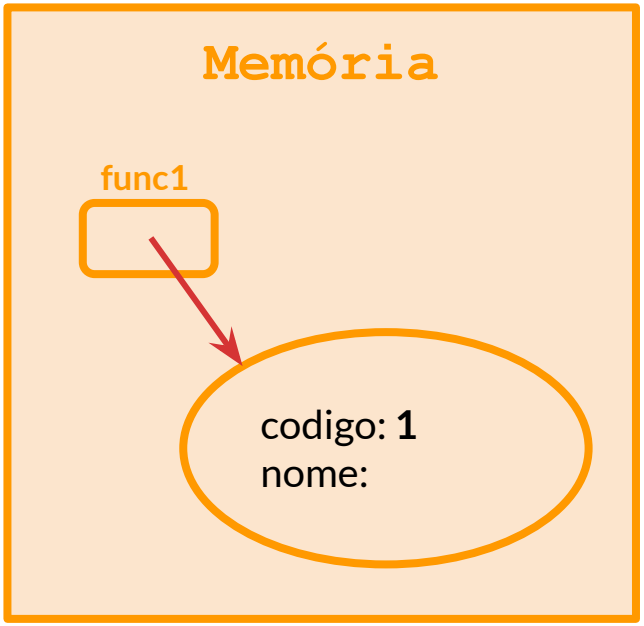
Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```



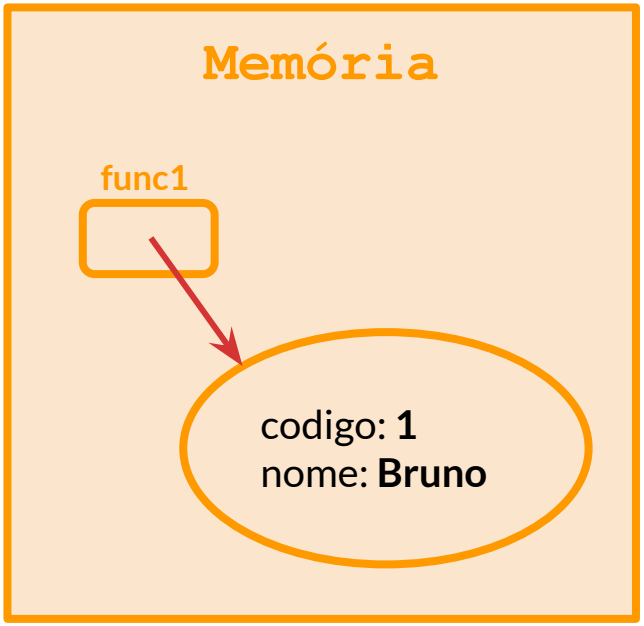
Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```



Instanciando Objetos

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

System.out.printf("%d - %s",
            func1.codigo, func1.nome);
```



Encapsulamento

Controlando o acesso aos atributos

Encapsulamento

- Cada atributo de um objeto pode possuir um modificador de acesso que indica qual a visibilidade do atributo para outras classes.

Encapsulamento

- Utilizaremos os seguintes modificadores de acesso para os atributos:
  - **public**: um atributo público pode ser acessado por qualquer outra classe que tenha acesso ao objeto em questão.
  - **private**: um atributo privado só é acessível dentro da própria classe.
  - **protected**: um atributo protegido só é acessível dentro da própria classe ou de qualquer outra classe que herde da classe em questão.

    Observação: herança será discutido em outro momento.

Encapsulamento

- Para impedirmos que nossos atributos tenham valores atribuídos ou lidos em desacordo com as regras que nos interessam, vamos alterar o modificador de acesso dos atributos da nossa classe Funcionario para private.
  - Com isso, os mesmos não poderão ser acessados diretamente no main.

```
public class Funcionario {
    private int codigo;
    private String nome;
}
```

Encapsulamento

- A alteração dos atributos para private impede que os mesmos sejam acessados no main.

- Para tornar este acesso possível, vamos criar os métodos acessores.
  - Teremos dois métodos para cada atributo.
    - Um método get para permitir ler o valor do atributo
    - Um método set para permitir alterar o valor do atributo

Encapsulamento

Método:	Utilizado para:
get	Obter o valor
set	Atribuir o valor

Encapsulamento

- O atributo **codigo** com assessores fica assim:

```
private int codigo;

public  int getCodigo() {
    return codigo;
}

public void setCodigo(int novoCodigo) {
    codigo = novoCodigo;
}
```

Encapsulamento

- Uma outra possibilidade seria:

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}

public void setCodigo(int codigo) {
    this.codigo = codigo;
}
```

Encapsulamento

- Uma outra possibilidade seria:

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}

public void setCodigo(int codigo)
    this.codigo = codigo;
}
```

- Como o nome do parâmetro do método **setCodigo** é o mesmo nome do atributo **codigo**, não é possível atribuir desta forma: **codigo = codigo;**

Encapsulamento

- Uma outra possibilidade seria:

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}

public void setCodigo(int codigo)
    this.codigo = codigo;
}
```

- Uma solução é usar a palavra reservada **this**, que indica “este objeto”. Logo, dentro do método **setCodigo**, **codigo** é o parâmetro enquanto **this.codigo** é o atributo, pois estamos acessando **codigo** do “this” e não do método.

Encapsulamento

- O atributo **nome** com assessores fica assim:

```
private String nome;

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}
```

Encapsulamento

- Código completo da classe Funcionario:

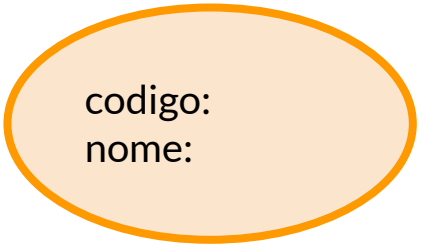
```
public class Funcionario {
    public  int getCodigo() {
        return this.codigo;
    }
    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    private int codigo;
    private String nome;
}
```

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                    func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento



- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.println("Código: " + func1.getCodigo() + "\nNome: " + func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(int codigo) {
    this.codigo = codigo;
}
```

codigo:
nome:

- Atribuindo 1 para o código do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.println("Código: " + func1.getCodigo() + "\nNome: " + func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(int codigo) {
    this.codigo = codigo;
}
```

codigo:
nome:

- Atribuindo 1 para o código do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");
                                                    func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(1) {
    this.codigo = 1;
}
```

codigo:
nome:

- Atribuindo 1 para o código do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");
                                                func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(1) {
    this.codigo = 1;
}
```

codigo:
nome:

- Atribuindo 1 para o código do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.println(func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(1) {
    this.codigo = 1;
}
```

codigo: **1**
nome:

- Atribuindo 1 para o código do funcionário.

Encapsulamento

codigo: **1**
nome:

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");
                                                    func1.getNome());
    }
}
```

- Atribuindo 1 para o código do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main
        Funcionario func
        func1.setCodigo(1);
        func1.setNome("Bruno");

    }
}
```

```
private String nome;

public void setNome(String nome) {
    this.nome = nome;
}
```

codigo: **1**
nome:

```
func1.getNome());
```

- Atribuindo Bruno para o nome do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main
        Funcionario func
        func1.setCodigo(1);
        func1.setNome("Bruno");

        func1.getNome());
    }
}
```

```
private String nome;

public void setNome("Bruno") {
    this.nome = "Bruno";
}
```

codigo: **1**
nome:

- Atribuindo Bruno para o nome do funcionário.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main
        Funcionario func
        func1.setCodigo(1);
        func1.setNome("Bruno");

    }
}
```

```
private String nome;

public void setNome("Bruno") {
    this.nome = "Bruno";
}
```

codigo: **1**
nome:

- Atribuindo Bruno para o nome do funcionário.

```
func1.getNome());
```

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main
        Funcionario func
        func1.setCodigo(1);
        func1.setNome("Bruno");

        func1.getNome());
    }
}
```

```
private String nome;

public void setNome("Bruno") {
    this.nome = "Bruno";
}
```

codigo: **1**
nome: **Bruno**

- Atribuindo Bruno para o nome do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        func1.getNome());
    }
}
```

- Atribuindo Bruno para o nome do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 =
        func1.setCodigo(1);
        func1.setNome("Brun

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 =
        func1.setCodigo(1);
        func1.setNome("Brun

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 =
        func1.setCodigo(1);
        func1.setNome("Brun

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public  int getCodigo() {
    return this.codigo;
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 =
        func1.setCodigo(1);
        func1.setNome("Brun

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public  int getCodigo() {
    return 1;
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

- Retornando o código do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

- Retornando o nome do funcionário.

Encapsulamento

- Novo código do main:

codigo: **1**
nome: **Bruno**

```
public class Main {
    public static void main(String[] args) {
        Funcionari
        func1.setC
        func1.setN

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private String nome;

public String getNome() {
    return nome;
}
```

- Retornando o nome do funcionário.

Encapsulamento

- Novo código do main:

codigo: **1**
nome: **Bruno**

```
public class Main {
    public static void main(String[] args) {
        Funcionari
        func1.setC
        func1.setN

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private String nome;

public String getNome() {
    return nome;
}
```

- Retornando o nome do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionari
        func1.setC
        func1.setN

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private String nome;

public String getNome() {
    return nome;
}
```

- Retornando o nome do funcionário.

Encapsulamento

- Novo código do main:

codigo: **1**
nome: **Bruno**

```
public class Main {
    public static void main(String[] args) {
        Funcionari
        func1.setC
        func1.setN

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private String nome;

public String getNome() {
    return "Bruno";
}
```

- Retornando o nome do funcionário.

Encapsulamento

codigo: **1**
nome: **Bruno**

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(1);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

- Retornando o nome do funcionário.

Qual a vantagem do uso do encapsulamento?

Encapsulamento

- Suponha que não seja desejado um número negativo como código de um funcionário:

```
private int codigo;

public  int getCodigo() {
   return this.codigo;
}
public void setCodigo(int codigo) {
   this.codigo = (codigo >= 0 ? codigo : codigo * -1);
}
```

Encapsulamento

- Suponha que não seja desejado um número negativo como código de um funcionário:

```
pri
pub
}
public void setCodigo(int codigo) {
    this.codigo = (codigo >= 0 ? codigo : codigo * -1);
}
```

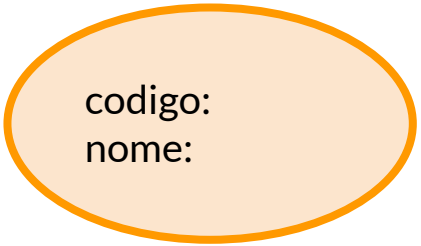
- Com o encapsulamento, é possível controlarmos o estado do objeto de forma a não possibilitar que o mesmo seja colocado em um estado inválido.

Encapsulamento

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                    func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento



- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento

- Novo código d

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(int codigo) {
    this.codigo = (codigo >= 0 ? codigo : codigo * -1);
}
```

codigo:
nome:

Encapsulamento

- Novo código d

```
private int codigo;

public void setCodigo(int codigo) {
    this.codigo = (codigo >= 0 ? codigo : codigo * -1);
}
```

codigo:
nome:

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento

- Novo código d

```
private int codigo;

public void setCodigo(-10) {
    this.codigo = (-10 >= 0 ? -10 : -10 * -1);
}
```

codigo:
nome:

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

Encapsulamento

- Novo código d

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(-10) {
    this.codigo = (-10 >= 0 ? -10 : -10 * -1);
}
```

codigo:
nome:

Encapsulamento

- Novo código d

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(-10) {
    this.codigo = (-10 >= 0 ? -10 : -10 * -1);
}
```

false

codigo:
nome:

Encapsulamento

- Novo código d

```
public class
    public st
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

```
private int codigo;

public void setCodigo(-10) {
    this.codigo = (-10 >= 0 ? -10 : -10 * -1);
}
```

10

codigo: **10**
nome:

Encapsulamento

codigo: **10**
nome:

- Novo código do main:

```
public class Main {
    public static void main(String[] args) {
        Funcionario func1 = new Funcionario();
        func1.setCodigo(-10);
        func1.setNome("Bruno");

        System.out.printf("%d - %s",
                func1.getCodigo(), func1.getNome());
    }
}
```

Exercícios

1) Crie uma classe para representar um aluno. Um aluno possui uma matrícula, um nome e um e-mail.

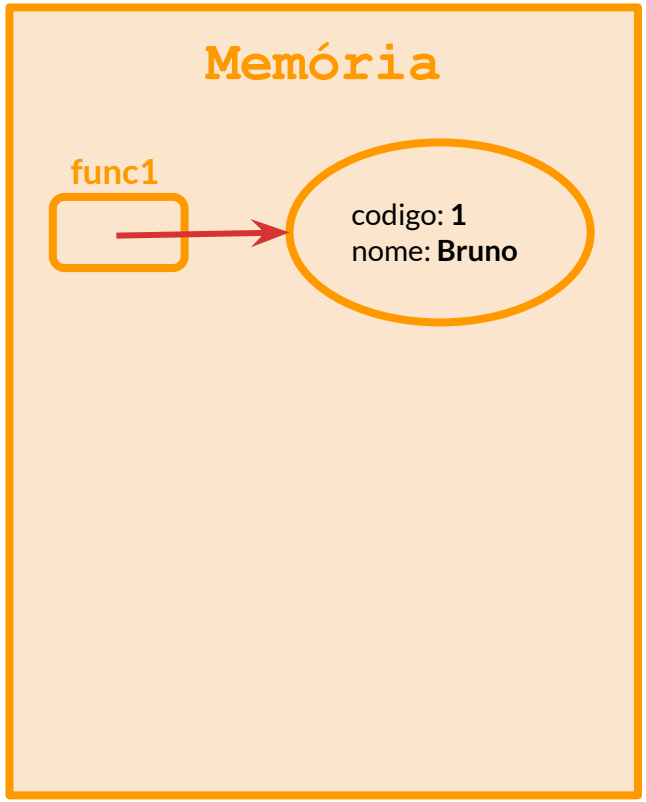
2) No projeto criado no exercício (1), altere o main para que sejam lidas as informações para três alunos. Após a leitura, imprima todas as informações de todos os alunos.

Trabalhando com Instâncias

Trabalhando com Múltiplas Instâncias e Múltiplas Referências

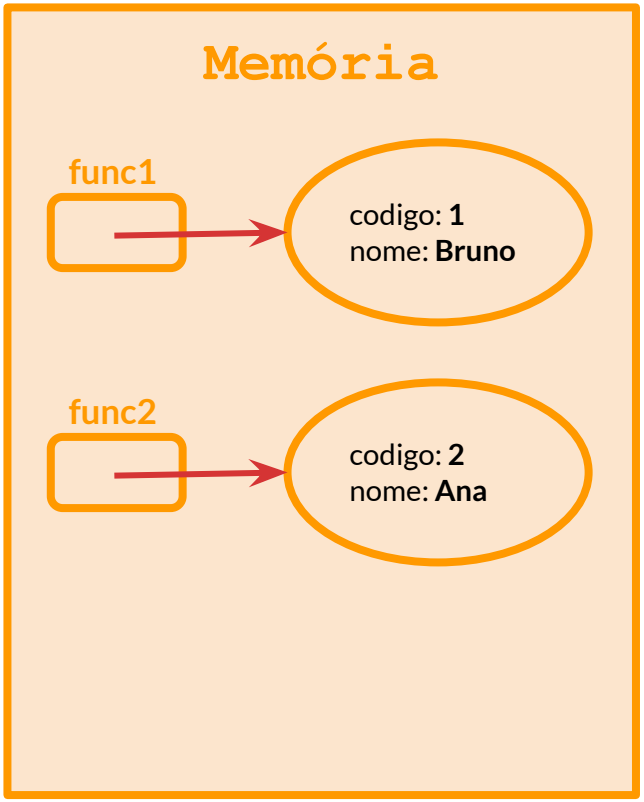
Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



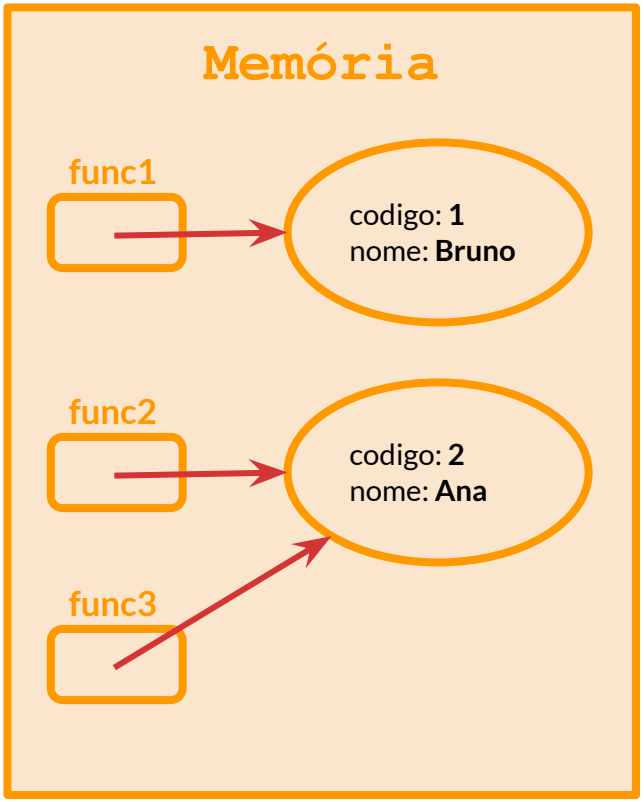
Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



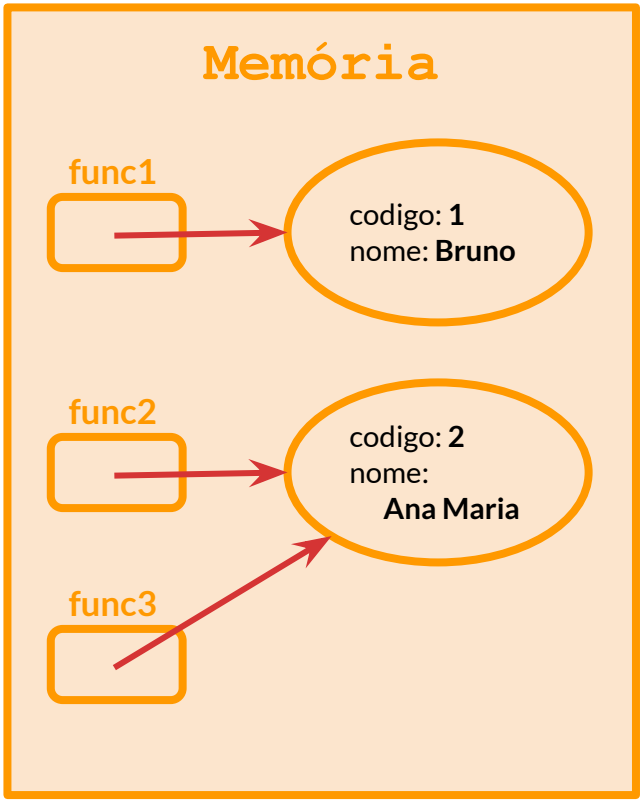
Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



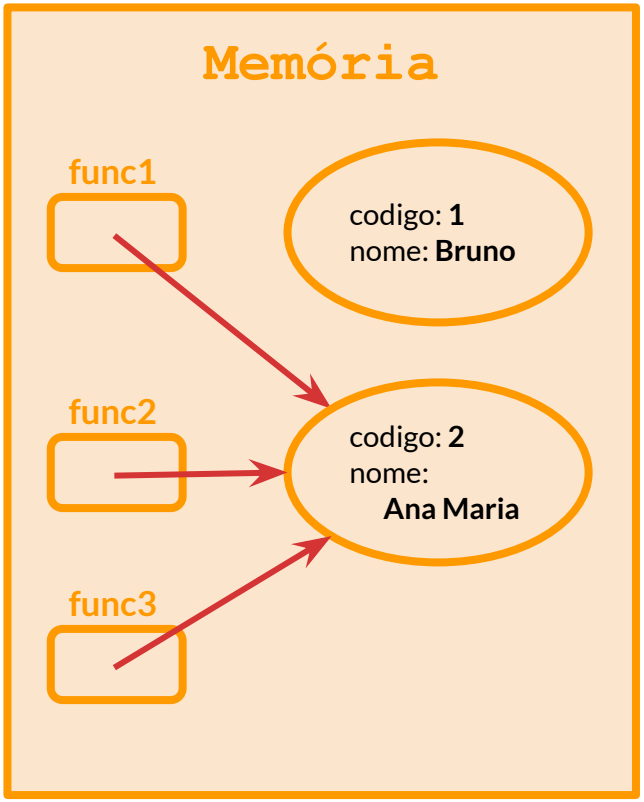
Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



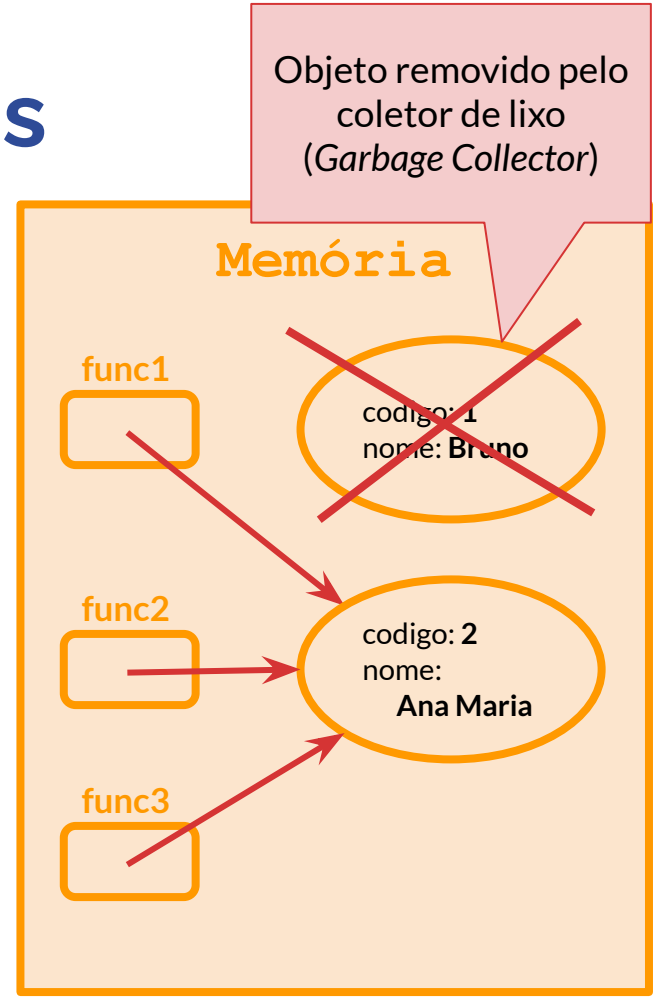
Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



Trabalhando com Instâncias

```
Funcionario func1 = new Funcionario();
func1.codigo = 1;
func1.nome = "Bruno";

Funcionario func2 = new Funcionario();
func2.codigo = 2;
func2.nome = "Ana";

Funcionario func3 = func2;

func3.nome = "Ana Maria";

func1 = func3;
```



Fim de material

Dúvidas?