



Programação

Getters && Setters
Por Carla Macedo

Construtores

- **Métodos especiais** que permitem passar valores a atributos de um objeto, durante a criação do mesmo
- Como o seu nome indica este tipo de método **constrói os objetos** e nesse instante pode definir quaisquer dos seus atributos
- Os objetos são criados dinamicamente durante a execução dos programas



Construtores

```
class Musica{
```

```
    String titulo;  
    double duracao;
```

```
    Musica(String t, double d){
```

```
        titulo = t;
```

```
        duracao = d;
```

```
    }
```

```
}
```

Exemplo de
um possível
método
construtor



Criar objetos

Os objetos criam-se com a palavra **new** seguida do construtor

Exemplo:

```
Musica m1 = new  
Musica("Yellow",3.8)
```

m1 é um
objeto da
classe
Musica



Construtor da Classe

Construtor da classe, não é necessário ser declarado pois toda a classe tem um construtor default de domínio público

```
public Pessoa{  
}
```

.....



Construtor

- Define como os atributos do objeto vão ser inicializados
- O nome do construtor deve ser exatamente o nome da Classe



Comportamento do Construtor



Construtor **inicializa** onde é chamado o método **números()**

```
public class inicializa {  
  
    inicializa(){  
        numeros();  
    }  
  
    public void numeros() {  
  
        int num = 5;  
        int num1 = 20;  
        System.out.println("A soma dos numeros é " + (num+num1));  
    }  
}
```

A classe principal vai chamar o Construtor **inicializa()** que por sua vez tem lá dentro o método **números()** e executa-o

```
public class Construtor {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        inicializa p = new inicializa();  
    }  
}
```



Se os atributos tiverem o modificador como Private, não conseguem ser acedidos em outras classes, para resolver este problema, utiliza-se os:

Getters e Setters



Getters – método que retorna/obtem o valor do atributo.

Setters – Método que atribui/modifica o valor do atributo

Utiliza-se a palavra **this** para imputar ao atributo o valor que é passado por parâmetro.

Declaração dos atributos da Classe



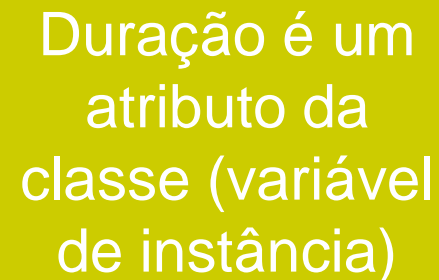
```
public classe Pessoa{  
private double altura;  
private string nome;  
private double peso;
```

```
.....
```

Getters e Setters

- **Getters**: São métodos de instância que tipicamente devolvem o valor de um atributo de um objeto (instância da classe).

```
public double getDuracao() {  
    return duracao;  
}
```

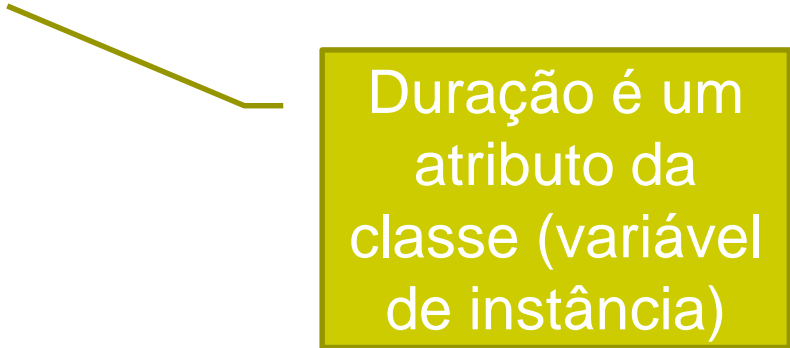
A yellow rectangular callout box with a black border and a black line pointing from the 'duracao' variable in the code above to the text inside.

Duração é um atributo da classe (variável de instância)

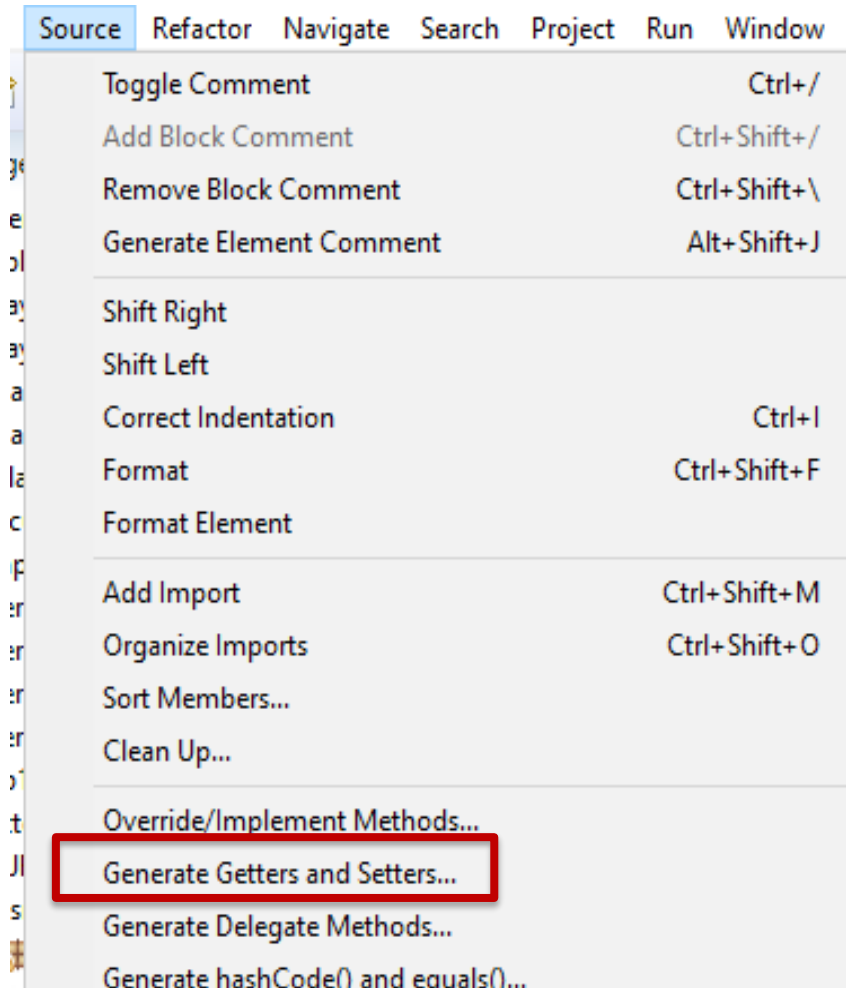
Getters e Setters

- **Setters**: São métodos de instância que tipicamente atribuem um valor, recebido por parâmetro, a um atributo de um objeto (instância da classe). Regra geral não têm retorno (void)

```
public void setDuracao(double duracao) {  
    this.duracao = duracao;  
}
```

A yellow rectangular callout box with a black border. A thin black line (pointer) originates from the left side of the box and points towards the 'this.duracao' property access in the code block above.

Duração é um atributo da classe (variável de instância)





Getters and Setters

```
public Double getAltura() {  
    return altura;  
}
```

```
public Double setAltura(Double altura) {  
    this.altura=altura;  
}
```



Aceder aos dados

```
...main(...){
```

```
//Instanciar a classe
```

```
Pessoa people = new Pessoa();
```

```
//atribui valor ao atributo
```

```
people.setAltura(1.60);
```

```
//mostrar os dados
```

```
System.out.println("tem altura"
```

```
+people.getAltura());
```



```
public static void main(String[] args) {  
    // TODO Auto-generated method stub
```

```
    Scanner sc = new Scanner(System.in);  
    System.out.print("numero: ");  
    int num = sc.nextInt();  
    System.out.print("numero: ");  
    int num1 = sc.nextInt();
```

```
    GetSet venda = new GetSet();
```

```
    venda.setA(num);  
    venda.setB(num1);
```

```
    System.out.println(venda.getA());  
    System.out.println(venda.getB());  
    System.out.println(venda.getA()+venda.getB());  
    System.out.println("total com getCal: 3"+venda.getCal());  
  
    sc.close();
```

```
public class GetSet {  
    private int a;  
    private int b;  
    private int cal;  
  
    public int getA() {  
        return a;  
    }  
  
    public void setA(int a) {  
        this.a = a;  
    }  
  
    public int getB() {  
        return b;  
    }  
  
    public void setB(int b) {  
        this.b = b;  
    }  
  
    public int getCal() {  
        return a + b;  
    }  
}
```


Instanciar a classe



Criar o objeto o nome **GetSet** = nome da classe

Venda (nome que quiserem dar)

```
GetSet venda = new GetSet();
```

F3 – ao selecionar o **GetSet** o Java mostra qual a classe de referencia



Exercício - Array

Programa que gere a entrega de 3 prêmios aleatórios.

Premio 1 – casa de férias

Prêmio 2 – Viagem

Prêmio 3 - Concerto



Exercício - Array

```
public class Premio{  
    String premios [] =new String[3];  
  
    Premio(){ //Método Construtor (serve para inicializar as  
                                                variáveis)  
        this.premios[0]="casa de ferias";  
        this.premios[1]="viagem";  
        this.premios[2]="Concerto";  
    }  
    public String sorteiaPremio(){  
        Random valor =new Random();  
        int posição = valor.nextInt(3);  
        return prémios[posição];  
    }  
}
```

Classe principal



.....

```
Premio p = new Premio();
```

```
System.out.println("o seu premio é: " + p.sorteioPremio());
```

Resumo



- Para aceder a atributos do tipo Private em outras classes, tem de criar Getters e Setters;
- Os Getters and Setters são do tipo public o que significa que são válidos fora da própria classe.



1. Escreva a classe Conta. Ela deve possuir número, saldo e limite, e deve ser possível creditar, debitar e transferir valores.
 - Não deve ser possível debitar um valor maior que o saldo;
2. Instancie 3 contas diferentes e utilize pelo menos uma vez cada uma das operações implementadas na questão anterior.



FIM

ORACLE®

