

# Programação Orientada a Objetos

## Classes, Métodos e Atributos

Professor Isaac

# Criando métodos

# Criação de métodos

- A sintaxe para criação de um método tem alguns itens obrigatórios e outros opcionais:

```
<tipo de acesso> <tipo de modificador> <tipo do retorno> <nomeDoMetodo> (<parâmetros>){  
    <comandos> // operações realizadas pelo método  
}
```

- Obrigatórios:** tipo do retorno, nome do método, parênteses, chaves e comandos
- Opcionais:** tipo de acesso, tipo de modificador, parâmetros

| Conta                           |           |
|---------------------------------|-----------|
| - numero: int                   | Atributos |
| - nome: String                  |           |
| - saldo: double                 |           |
| + debitarSaldo(double): boolean | Métodos   |

# Encapsulamento - Modificadores de Acesso

---

- *public:*

- Indica que um método ou atributo é acessível a outras funções e funções-membro de outras classes.

- *private:*

Torna um membro de dados ou uma função-membro acessível apenas a funções-membro da classe.

- *protected:*

Torna o membro acessível às classes do mesmo pacote ou através de herança

# Tipo de Retorno

- O retorno é a saída do método, o que irá retornar quando voltar para quem a chamou o método.
- O tipo de retorno pode ser um **tipo primitivo** ou **uma classe**

# Tipo de métodos quanto ao retorno e aos parâmetros

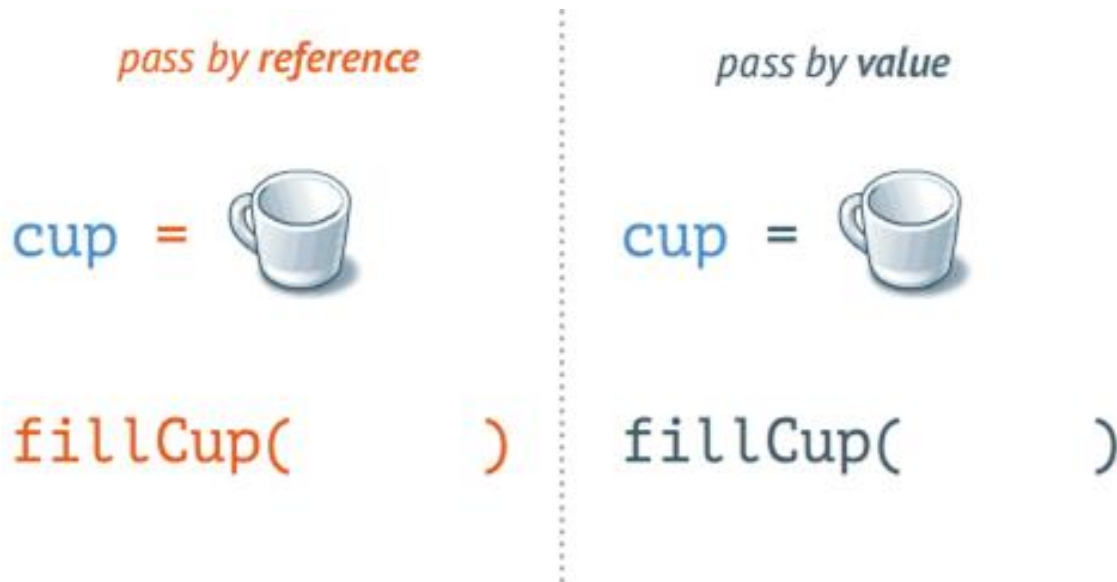
- Método **SEM** retorno e **SEM** parâmetros
- Método **COM** retorno e **SEM** parâmetros
- Método **SEM** retorno e **COM** parâmetros
- Método **COM** retorno e **COM** parâmetros

# Java - Passagem de Argumentos

Java não permite que o programador escolha entre passar por valor ou referência.

Variáveis dos **tipos primitivos** sempre são passadas por **valor**.

Os **objetos** são passados por **referência**, sendo que as próprias referências são passadas por valor.



# Método SEM retorno e SEM parâmetros

- Os métodos que não tem valor de retorno e também não recebem parâmetros utilizam a palavra “*void*” para indicar sem retorno (ou retorno vazio) e os parênteses vazios para indicar sem parâmetros:

- Sintaxe:**

```
void <nomeDoMétodo> ( ) {  
    <comandos>  
}
```

- Exemplo:**

```
void printSaldo(){  
    System.out.println(this.saldo);  
}
```



# Método COM retorno e SEM parâmetros

- O **retorno** de um método é o valor que é retornado **desse método para aquele que o chamou**
- No método para retornar o valor deve ser usado o comando “*return*”:

- **Sintaxe:**

```
<tipo> <nomeDoMétodo> ( ) {  
    <comandos>  
    return <valor> ;  
}
```

- **Exemplo:**

```
int getSaldo(){  
    return this.saldo;  
}
```

# Método SEM retorno e COM parâmetros

- Os **parâmetros** de um método são os valores que são enviados **do método que chamou para o outro método**

- Sintaxe:**

```
void <nomeDoMétodo> ( <tipo1> <nome1>, <tipo2> <nome2>, ... ,  
                    <tipoN> <nomeN>) {  
    <comandos>  
}
```

- Exemplo:**

```
void setSaldo(int saldo){  
    this.saldo = saldo;  
}
```

# Método COM retorno e COM parâmetros

- O **retorno** de um método é o valor que é retornado **desse método para aquele que o chamou** usando o comando **return**
- Os **parâmetros** de um método são os valores que são enviados **do método que chamou para o outro método**

- **Sintaxe:**

```
<tipo> <nomeDoMétodo> (<tipo1> <nome1>, ... , <tipoN> <nomeN>) {  
    <comandos>  
    return <valor> ;  
}
```

- **Exemplo:**

```
public boolean debitarSaldo(double quantidade) {  
    if (this.saldo >= quantidade) {  
        this.saldo = this.saldo - quantidade;  
        return true;  
    } else {  
        return false;  
    }  
}
```

# Importante

- **Apenas um** valor pode ser retornado por um método, no entanto objetos de classes podem ser retornados.
- **Um ou mais parâmetros** podem ser recebidos pelo método
- **Qualquer método** também pode chamar outro método passando argumentos e recebendo retorno

# Getters e Setters

- O uso indiscriminado de métodos Getters e Setters torna o encapsulamento sem sentido.
- Os atributos são definidos como privados para evitar o acesso de outras classes, mas os Getters e Setters liberam esse acesso.
- Uma boa prática é utilizar os Construtores para atribuir valor inicial aos atributos no lugar de usar Setters

# Método Construtor

# Métodos Construtor

- Utilizado para construir (instanciar) o objeto
- Deve ter o mesmo nome da classe
- Único método que NÃO deve usar tipo de retorno ou “void” na sua criação
- Toda classe criada tem um método construtor padrão sem parâmetros: <NomeDaClasse> ( )
- Outros métodos construtores podem ser criados com parâmetros

# Criação de Método Construtor

- Sintaxe:

```
<tipo de acesso> <NomeDaClasse> (<parâmetros>) {  
    <comandos> // operações realizadas pelo método  
}
```

- **Exemplo classe Conta:**

```
— public Conta (int numero) {  
    this.numero = numero;  
}
```



# Referenciando membros do objeto atual com a referência `this`

- Qualquer objeto pode acessar uma referência dele mesmo com a palavra chave `this`.
- Métodos não-estáticos utilizam implicitamente `this` ao referenciar variáveis de instância do objeto e outros métodos.
- Quando um método contém um parâmetro ou variável local com o mesmo nome de um campo da classe, utilize a referência `this` se desejar acessar o atributo da classe. Caso contrário, o parâmetro ou variável local do método será referenciado.

# Construtores-padrão e sem argumentos

- Toda classe deve ter pelo menos um construtor.
- Se nenhum construtor for declarado, o compilador criará um construtor-padrão que não recebe nenhum argumento e inicializa os atributos da seguinte forma:
  1. De acordo com seus valores iniciais especificados nas suas declarações;ou
  2. De acordo com seus valores-padrão:
    - **zero** para tipos numéricos primitivos
    - **false** para valores boolean
    - **null** para referências

# Exemplo Classe Conta

| Conta                           |
|---------------------------------|
| - numero: int                   |
| - nome: String                  |
| - saldo: double                 |
| + Conta(int)                    |
| + getNumero(): int              |
| + getNome(): String             |
| + getSaldo(): double            |
| + setNome(int): void            |
| + debitarSaldo(double): boolean |

# Exemplo Classe Conta

| Conta                           |
|---------------------------------|
| - numero: int                   |
| - nome: String                  |
| - saldo: double                 |
| + Conta(int)                    |
| + getNumero(): int              |
| + getNome(): String             |
| + getSaldo(): double            |
| + setNome(int): void            |
| + debitarSaldo(double): boolean |

```
public class Conta {  
    private int numero;  
    private String nome;  
    private double saldo;  
  
    public Conta(int numero) {  
        this.numero = numero;  
        this.saldo = 0;  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
}
```

# Exemplo Classe Conta

| Conta                           |
|---------------------------------|
| - numero: int                   |
| - nome: String                  |
| - saldo: double                 |
| + Conta(int)                    |
| + getNumero(): int              |
| + getNome(): String             |
| + getSaldo(): double            |
| + setNome(int): void            |
| + debitarSaldo(double): boolean |

```
public class Conta {  
    private int numero;  
    private String nome;  
    private double saldo;  
  
    public Conta(int numero) {  
        this.numero = numero;  
        this.saldo = 0;  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
}
```

**Construtor**

# Exemplo Classe Conta

## Conta

- numero: int

- nome: String

- saldo: double

+ Conta(int)

+ getNumero(): int

+ getNome(): String

+ getSaldo(): double

+ setNome(int): void

+ debitarSaldo(double): boolean

```
public class Conta {  
    private int numero;  
    private String nome;  
    private double saldo;  
  
    public Conta(int numero) {  
        this.numero = numero;  
        this.saldo = 0;  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
}
```

- Métodos **get** não recebe parâmetros
- Retorna o valor do atributo

# Exemplo Classe Conta

## Conta

- numero: int

- nome: String

- saldo: double

+ Conta(int)

+ getNumero(): int

+ getNome(): String

+ getSaldo(): double

+ setNome(int): void

+ debitarSaldo(double): boolean

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

- Métodos **set** recebe o parâmetro do atributo.
- Não retorna valor.

```
public boolean debitarSaldo(double quantidade) {  
    if (this.saldo >= quantidade) {  
        this.saldo = this.saldo - quantidade;  
        return true;  
    } else {  
        return false;  
    }  
}
```

# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoal = new Conta(1298);  
  
        pessoal.setNome("José");  
  
        double saldo = pessoal.getSaldo();  
        int numero = pessoal.getNumero();  
        String nome = pessoal.getNome();  
  
        System.out.println("Nome: " + nome + " | Saldo: " + saldo);  
    }  
}
```

➤ Atribui valor nos parâmetros do construtor quando instancia um objeto.



# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoal = new Conta(1298);  
  
        pessoal.setNome("José");  
  
        double saldo = pessoal.getSaldo();  
        int numero = pessoal.getNumero();  
        String nome = pessoal.getNome();  
  
        System.out.println("Nome: " + nome + "| Saldo: " + saldo);  
    }  
}
```

Saída - Aula03 (run) x



run:



Nome: José| Saldo: 0.0



CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Sobrecarga de Construtor

# Sobrecarga de Construtor

- Podemos criar vários construtores que recebem quantidades e tipos de parâmetros diferentes.
- **Exemplos:**

```
public Conta (int numero) {  
    this.numero = numero;  
    this.saldo = 0;  
}  
public Conta (int numero, String nome) {  
    this.numero = numero;  
    this.nome = nome;  
    this.saldo = 0;  
}  
public Conta (int numero, String nome, double saldo) {  
    this.numero = numero;  
    this.nome = nome;  
    this.saldo = saldo;  
}
```

# Exemplo Classe Conta

| Conta                           |
|---------------------------------|
| - numero: int                   |
| - nome: String                  |
| - saldo: double                 |
| + Conta()                       |
| + Conta(int)                    |
| + Conta(int, String)            |
| + Conta(int, String, double)    |
| + getNumero(): int              |
| + getNome(): String             |
| + getSaldo(): double            |
| + setNome(int): void            |
| + debitarSaldo(double): boolean |

- **Sobrecarga de Construtores.**
- **A classe possui 4 construtores.**

# Exemplo Classe Conta

## Conta

- numero: int

- nome: String

- saldo: double

+ Conta()

+ Conta(int)

+ Conta(int, String)

+ Conta(int, String, double)

+ getNumero(): int

+ getNome(): String

+ getSaldo(): double

+ setNome(int): void

+ debitarSaldo(double): boolean

```
public class Conta {  
    private int numero;  
    private String nome;  
    private double saldo;  
  
    public Conta() {  
        this.saldo = 0;  
    }  
  
    public Conta(int numero) {  
        this.numero = numero;  
        this.saldo = 0;  
    }  
  
    public Conta(int numero, String nome) {  
        this.numero = numero;  
        this.nome = nome;  
    }  
  
    public Conta(int numero, String nome, double saldo) {  
        this.numero = numero;  
        this.nome = nome;  
        this.saldo = saldo;  
    }  
}
```

# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoal = new Conta();  
        Conta pessoa2 = new Conta(1298);  
        Conta pessoa3 = new Conta(1299, "João");  
        Conta pessoa4 = new Conta(1300, "Maria", 7000.00);  
  
        pessoal.setNome("José");  
        pessoa2.setNome("Ana");  
  
        double saldo = pessoa3.getSaldo();  
        int numero = pessoa3.getNumero();  
        String nome = pessoa3.getNome();  
  
        System.out.println("Nome: " + nome + "| Saldo: " + saldo + "| N: " + numero);  
    }  
}
```

➤ Atribui valor nos parâmetros dos construtores, podendo usar qualquer um dos 4 construtores.

# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoal = new Conta();  
        Conta pessoa2 = new Conta(1298);  
        Conta pessoa3 = new Conta(1299, "João");  
        Conta pessoa4 = new Conta(1300, "Maria", 7000.00);  
  
        pessoal.setNome("José");  
        pessoa2.setNome("Ana");  
  
        double saldo = pessoa3.getSaldo();  
        int numero = pessoa3.getNumero();  
        String nome = pessoa3.getNome();  
  
        System.out.println("Nome: " + nome + "| Saldo: " + saldo + "| N: " + numero);  
    }  
}
```

Saída - Aula03 (run) x



run:



Nome: João| Saldo: 0.0| N: 1299



CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoa1 = new Conta();  
        Conta pessoa2 = new Conta(1298);  
        Conta pessoa3 = new Conta(1299, "João");  
        Conta pessoa4 = new Conta(1300, "Maria", 7000.00);  
  
        pessoa1.setNome("José");  
        pessoa2.setNome("Ana");  
  
        double saldo = pessoa4.getSaldo();  
        int numero = pessoa4.getNumero();  
        String nome = pessoa4.getNome();  
  
        System.out.println("Nome: " + nome + "| Saldo: " + saldo + "| N: " + numero);  
    }  
}
```



# Instanciando um Objeto da Classe Conta

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Conta pessoa1 = new Conta();  
        Conta pessoa2 = new Conta(1298);  
        Conta pessoa3 = new Conta(1299, "João");  
        Conta pessoa4 = new Conta(1300, "Maria", 7000.00);  
  
        pessoa1.setNome("José");  
        pessoa2.setNome("Ana");  
  
        double saldo = pessoa4.getSaldo();  
        int numero = pessoa4.getNumero();  
        String nome = pessoa4.getNome();  
  
        System.out.println("Nome: " + nome + "| Saldo: " + saldo + "| N: " + numero);  
    }  
}
```

**Saída - Aula03 (run)** x



run:



Nome: Maria| Saldo: 7000.0| N: 1300

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Método toString()

# Método toString()

- Usado quando precisamos mostrar os atributos de um objeto no formato desejado.
- Quando esse método não existe na classe e tentamos mostrar todo o objeto aparece um endereço de memória.

Exemplo: projtelevisao.Televisao@15db9742

- Para corrigir este problema devemos reescrever o método `toString()` dentro da classe.
- Este método sempre **retorna** uma **String** e será automaticamente chamado para mostrar detalhes do objeto quando usamos o método `System.out.println` ou outros método que mostra ou imprime um objeto.

# Exemplo **sem toString**:

## Mostra o endereço de memória

```
public class Principal {  
    public static void main(String[] args) {  
        Televisao tv1 = new Televisao(22, 10, false);  
        System.out.println("tv1: " + tv1);  
    }  
}
```

Saída

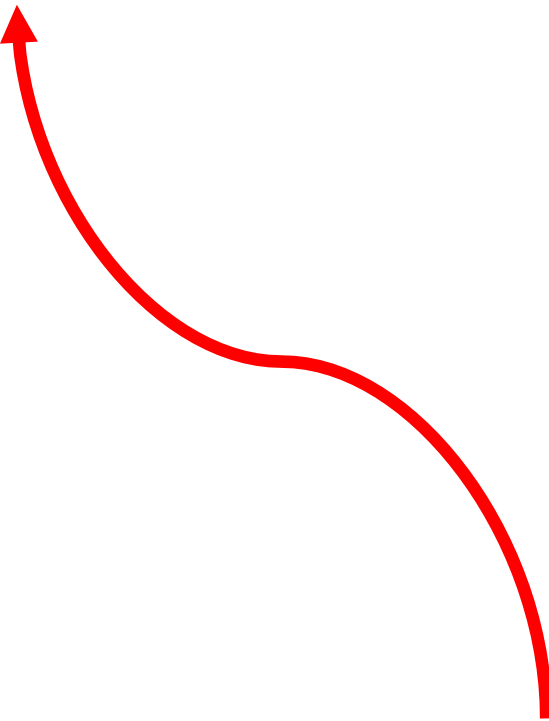
run:

tv1: projtelevisao.Televisao@15db9742

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Exemplo 1: Criando o método **toString()** para a classe Televisao

```
public String toString() {  
    return "Televisao{" + "canal=" + canal + ", som=" + som + ", ligada=" + ligada + '}';  
}
```



| Televisao                        |
|----------------------------------|
| - canal: int                     |
| - som: int                       |
| - ligada: boolean                |
| + Televisao ( )                  |
| + Televisao (int, int )          |
| + Televisao (int, int, boolean ) |
| + getCanal ( ): int              |
| + setCanal (int ): void          |
| + getSom( ): int                 |
| + setSom(int ): void             |
| + getLigada( ): boolean          |
| + setLigada(boolean): void       |
| + toString(): String             |

## Exemplo 1: usando o método **toString()** criado na classe Televisão

```
6 public class Principal {
7     public static void main(String[] args) {
8         // TODO code application logic here
9         Televisao tv1 = new Televisao( );
10        Televisao tv2 = new Televisao(8, 15);
11        Televisao tv3 = new Televisao(22, 10, true);
12        System.err.println(tv1);
13        System.err.println(tv2);
14        System.err.println(tv3);
15    }
16 }
```

O objeto invoca o método **toString()**

Find:  Previous Next Select



```
run:
Televisao{canal=0, som=0, ligada=false}
Televisao{canal=8, som=15, ligada=false}
Televisao{canal=22, som=10, ligada=true}
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Exemplo 2: Modificando o método toString() para a classe Televisao

```
public String toString() {  
    return " Canal = " + canal + "\n Som = " + som + "\n Ligada = " + ligada;  
}
```

## Exemplo 2: usando o método **toString()** modificado da classe Televisão

```
6 public class Principal {
7     public static void main(String[] args) {
8         // TODO code application logic here
9         Televisao tv1 = new Televisao( );
10        Televisao tv2 = new Televisao(8, 15);
11        Televisao tv3 = new Televisao(22, 10, true);
12        System.err.println("tv3:\n" + tv1);
13    }
14 }
```

Find:  Previous Next Select

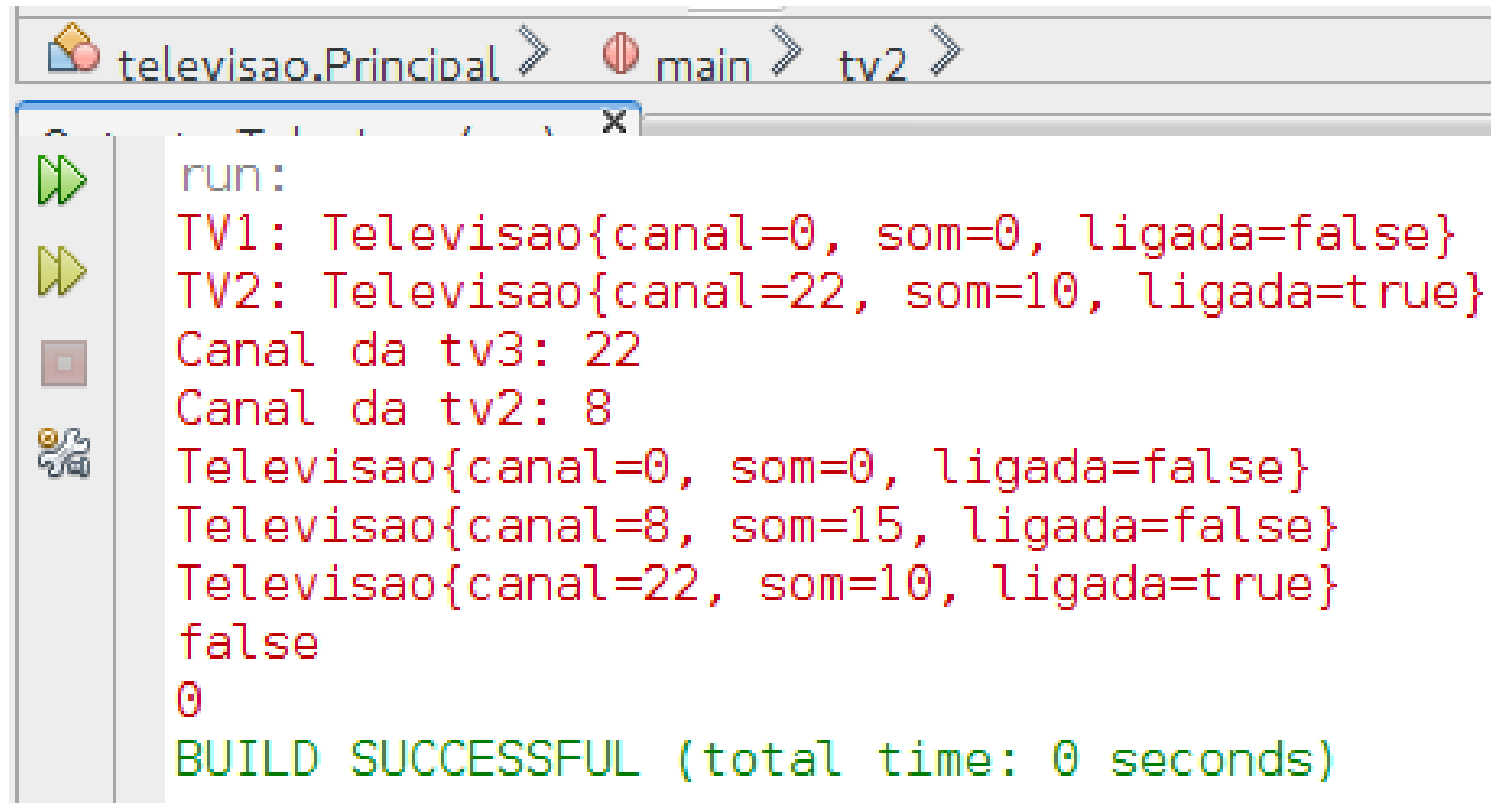
```
run:
tv3:
    Canal = 0
    Som = 0
    Ligada = false
BUILD SUCCESSFUL (total time: 0 seconds)
```



## Exemplo 3: usando os método **toString()**, getters e **setters** da classe Televisão

```
6 public class Principal {
7     public static void main(String[] args) {
8         // TODO code application logic here
9         Televisao tv1 = new Televisao( );
10        Televisao tv2 = new Televisao(8, 15);
11        Televisao tv3 = new Televisao(22, 10, true);
12        System.err.println("TV1: " + tv1);
13        System.err.println("TV2: " + tv3);
14        System.err.println("Canal da tv3: " + tv3.getCanal());
15        System.err.println("Canal da tv2: " + tv2.getCanal());
16        tv1.setLigada(true);
17        tv1.setLigada(false);
18        System.err.println(tv1);
19        System.err.println(tv2);
20        System.err.println(tv3);
21        System.err.println(tv1.getLigada());
22        System.err.println(tv1.getCanal());
23    }
24 }
```

## Exemplo 3: usando os método **toString**, getters e **setters** da classe Televisão



```
run:
TV1: Televisao{canal=0, som=0, ligada=false}
TV2: Televisao{canal=22, som=10, ligada=true}
Canal da tv3: 22
Canal da tv2: 8
Televisao{canal=0, som=0, ligada=false}
Televisao{canal=8, som=15, ligada=false}
Televisao{canal=22, som=10, ligada=true}
false
0
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Exercícios

# Exercício 1

- Implemente em Java o diagrama de classes abaixo:

| Data                  |
|-----------------------|
| - dia: int            |
| - mes: int            |
| - ano: int            |
| + Data(int, int, int) |
| + getDia(): int       |
| + getMes(): int       |
| + getAno(): int       |
| + setDia(int): void   |
| + setMes(int): void   |
| + setAno(int): void   |
| + toString(): String  |

# Exercício 2

- Implemente alguns métodos:
  - Crie o método `verificaMes()` para verificar se o mês é valido para o mês escolhido no construtor.
  - Crie o método `verificaDia()` para verificar se o dia é valido para o mês escolhido no construtor.
  - Não permitir valores inválidos nos métodos `setMes` e `setDia` .

| Data                    |
|-------------------------|
| - dia: int              |
| - mes: int              |
| - ano: int              |
| + Data(int, int, int)   |
| + verificaDia(int): int |
| + verificaMes(int): int |
| + getDia(): int         |
| + getMes(): int         |
| + getAno(): int         |
| + setDia(int): void     |
| + setMes(int): void     |
| + setAno(int): void     |
| + toString(): String    |

# Resposta Exercício 1 e 2

```
public class Data {
    private int dia;
    private int mes;
    private int ano;

    public Data(int dia, int mes, int ano) {
        this.mes = verificaMes(mes);
        this.ano = ano;
        this.dia = verificaDia(dia);
    }

    private int verificaDia(int testDia){

        int diasPorMes[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        // verifica se day está no intervalo para mes
        if ( testDia > 0 && testDia <= diasPorMes[ mes ] )
            return testDia;

        // verifica ano bissexto
        if ( mes == 2 && testDia == 29 &&
            ( ano % 400 == 0 || ( ano % 4 == 0 && ano % 100 != 0 ) ) )
            return testDia;

        System.out.printf( "Invalid day (%d) set to 1.", testDia );
        return 1;
    }
}
```

# Resposta Exercício 1 e 2

```
private int verificaMes(int mes){
    if ( mes > 0 && mes <= 12 ) // valida month
        return mes;
    else // month é inválido
    {
        System.out.printf("Invalid month (%d) set to 1.", mes );
        return 1; // mantém objeto em estado consistente
    }
}

public int getDia() {
    return dia;
}

public void setDia(int dia) {
    this.dia = dia;
}

public int getMes() {
    return mes;
}

public void setMes(int mes) {
    this.mes = ( ( mes >= 1 && mes <= 12 ) ? mes : 1 );
}
```

# Resposta Exercício 1 e 2

```
public int getAno() {  
    return ano;  
}  
  
public void setAno(int ano) {  
    this.ano = ano;  
}  
  
@Override  
public String toString() {  
    return "Data{" + "dia=" + dia + ", mes=" + mes + ", ano=" + ano + '}';  
}  
}
```



# Exercício 3

- Implemente vários construtores na classe Data:

| Data                    |
|-------------------------|
| - dia: int              |
| - mes: int              |
| - ano: int              |
| + Data(int)             |
| + Data(int, int)        |
| + Data(int, int, int)   |
| + verificaDia(int): int |
| + verificaMes(int): int |
| + getDia(): int         |
| + getMes(): int         |
| + getAno(): int         |
| + setDia(int): void     |
| + setMes(int): void     |
| + setAno(int): void     |
| + toString(): String    |