Revisão

+

Faculdade Mauricio de Nassau

Linguagem de Programação II

Curso de Engenharia de Telecomunicações

Programação Orientada a Objetos

- O que é POO?
  - É um paradigma de programação de computadores onde se usam classes e objetos, criados a partir de modelos (**mundo real**)
- Vantagens
  - Facilidade de Manutenção
  - Maior reuso

Principais Características (JAVA)

- Orientada a objetos
  - Java é uma linguagem puramente orientada a objetos
- Sem Ponteiros
  - Java não possui ponteiros, isto é, não permite a manipulação direta de endereços de memória

Principais Características (JAVA)

- Coletor de lixo (***Garbage Collector***)
  - Possui um mecanismo automático de gerenciamento de memória
- Permite ***Multithreading***
  - Múltiplas rotinas concorrentemente

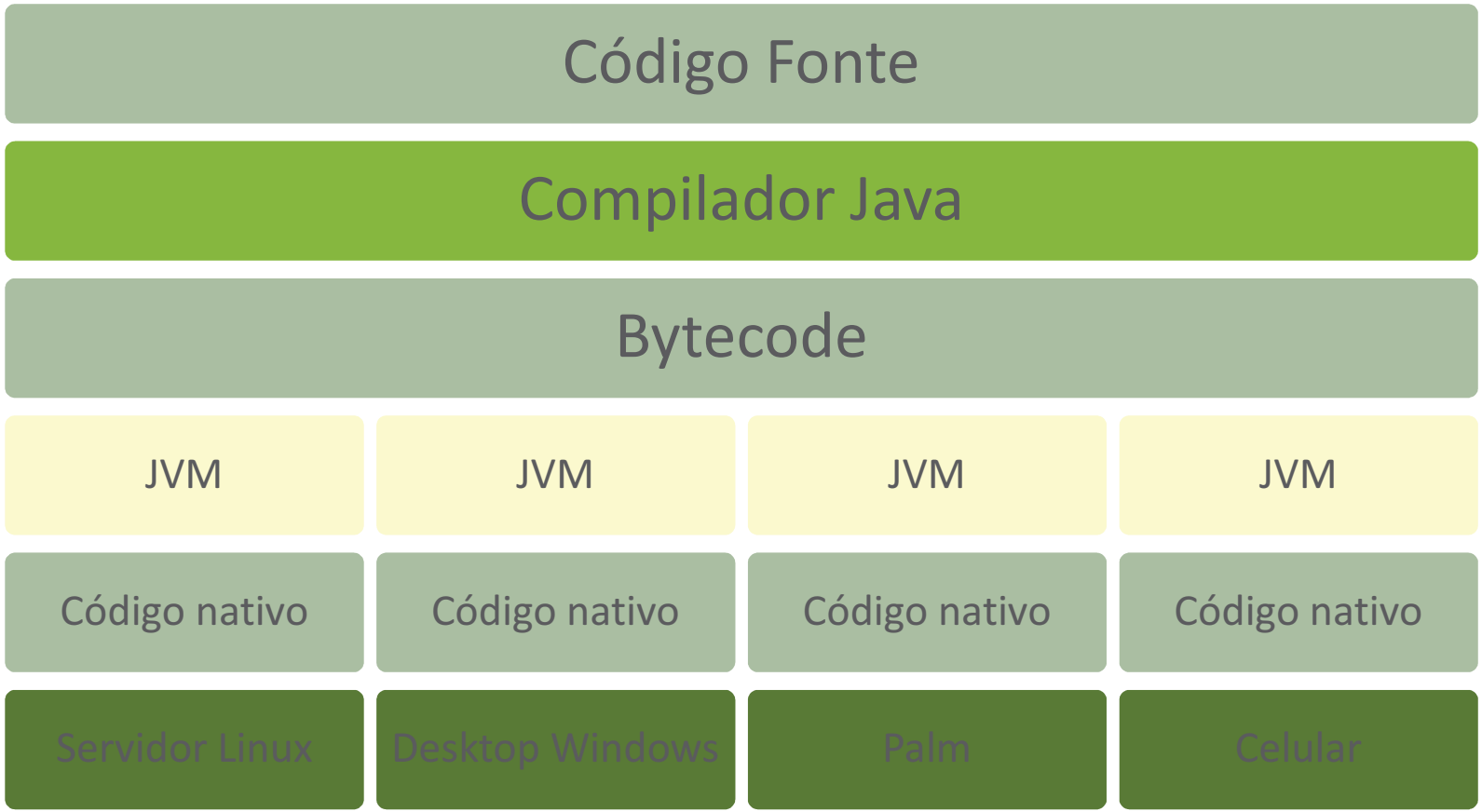
Principais Características (JAVA)

- "Independente de plataforma"
  - Programas Java são compilados para uma forma intermediária (*bytecodes*)
- Tratamento de exceções
  - Permite o tratamento de situações excepcionais
  - Possui exceções embutidas e permite a criação de novas exceções

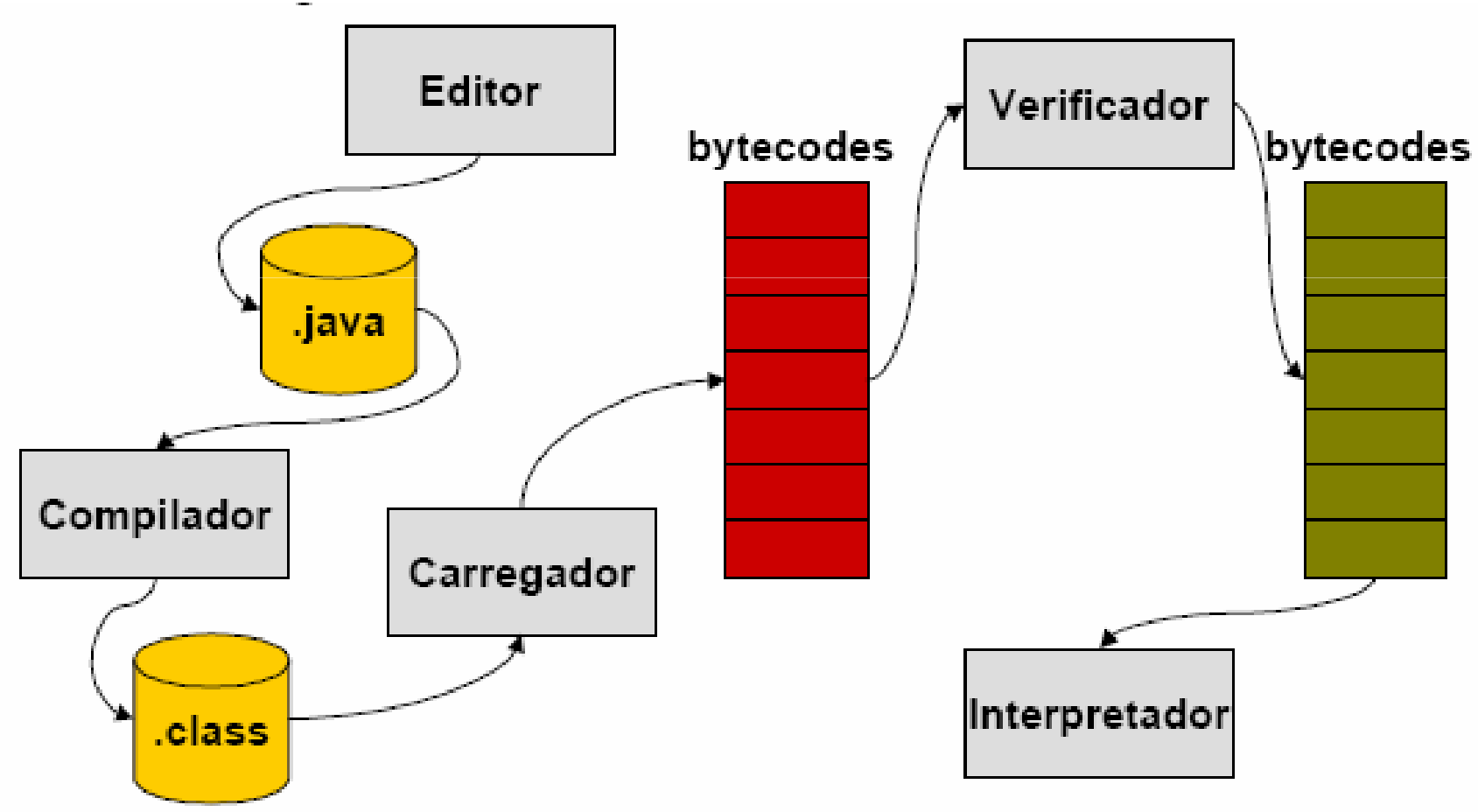
Principais Características (JAVA)

- Maquina Vitual Java (JVM)
  - Utiliza o conceito de máquina virtual
  - Camada responsável por interpretar os bytecodes

Compilador e Interpretador



Fases de um programa Java



Fases de um programa Java

- Fase 1 (**Edição**): Consiste em editar um arquivo com código em Java e salvá-lo com a extensão (**.java**)
- Fase 2 (**Compilação**): O compilador Java traduz (**.java => .class**)

Fases de um programa Java

- Fase 3 (**Carga**): Carrega o programa na memória antes de ser executado. Carregador de classe, pega o arquivo(s) **.class** que contém os ***bytecodes***
- Fase 4 (**Verificação**): O verificador assegura que os ***bytecodes*** são válidos e não violam as restrições de segurança de Java

Fases de um programa Java

- Fase 5 (**Execução**): A JVM máquina virtual Java (Interpretador) interpreta (em tempo de execução) o programa, realizando assim a ação especificada pelo programa

Estrutura Básica

- Objetos e Classes
- Pacotes
- Atributos
- Métodos

Objetos

- Um objeto é a materialização da classe, e assim pode ser usado para representar dados e executar operações.

```
Conta conta1 = new Conta(“02345”,1000);
```

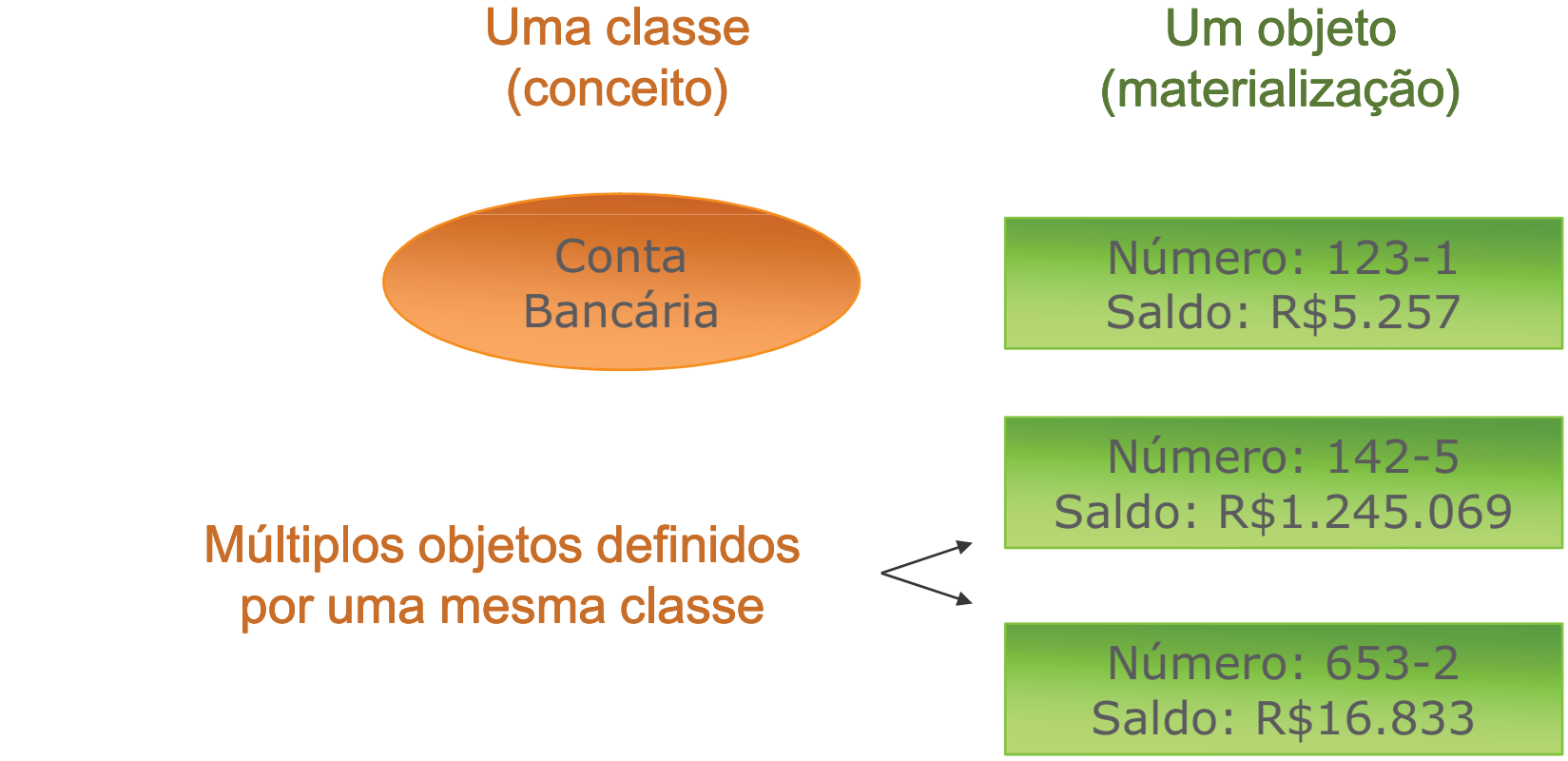
Objetos

- Um objeto possui:
  - Identidade: permite destingui-lo de outros
  - Estados: características
  - Comportamentos: o que pode ser feito com ele (ou nele)
- Por exemplo:
  - Os estados de uma conta bancária são o seu número e o seu saldo
  - Os comportamentos atribuídos a uma conta bancária são a habilidade de realizar depósitos e saques
- Note que o comportamento de um objeto pode modificar seus estados

Classes

- São estruturas das linguagens de programação orientadas a objetos para representar determinado modelo do **mundo real**
- Um Objeto é definido por uma classe
  - Características (**atributos**)
  - Comportamentos (**métodos**)

Objetos x Classes



Pacotes

- É um mecanismo para agrupar classes de finalidades afins ou de uma mesma aplicação.
- Facilita a organização conceitual das classes.
- Usamos a declaração ***import*** para acessar essas classes.

```
import java.lang.*;
```

Pacotes

- Usamos a declaração ***import*** para acessar essas classes.

```
import java.lang.*;
```

- Deve-se inserir a diretiva ***package*** com o nome do pacote para determinar o pacote a qual a classe pertence.

```
package java.lang;
```

Atributos

- Determinam as características do objeto
- Os modificadores são opcionais
- Vários atributos podem ser declarados na mesma linha
- Um atributo pode ser inicializado na declaração

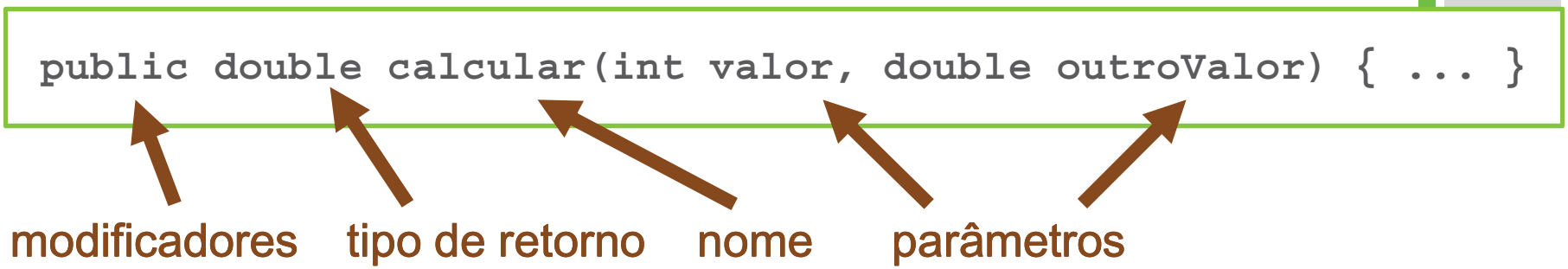
Exemplos

```
class Lapis {
      String cor;
}
```

```
class Cadeira {
      int numeroPernas;
      String fabricante;
}
```

```
class Conta{
      String numero;
      double saldo;
}
```

Métodos



- Operações que realizam ações ou modificam o objeto responsável pela sua execução
- O corpo do método determina o comportamento
- E também pode conter declaração de variáveis
  - Cuja existência e valores são válidos somente dentro do método em que são declaradas.

Exemplo (return)

```
class Conta{
        String numero;
        double saldo;

        String getNumero() {
                return numero;
        }

        double getSaldo() {
                return saldo;
        }

        ...
}
```

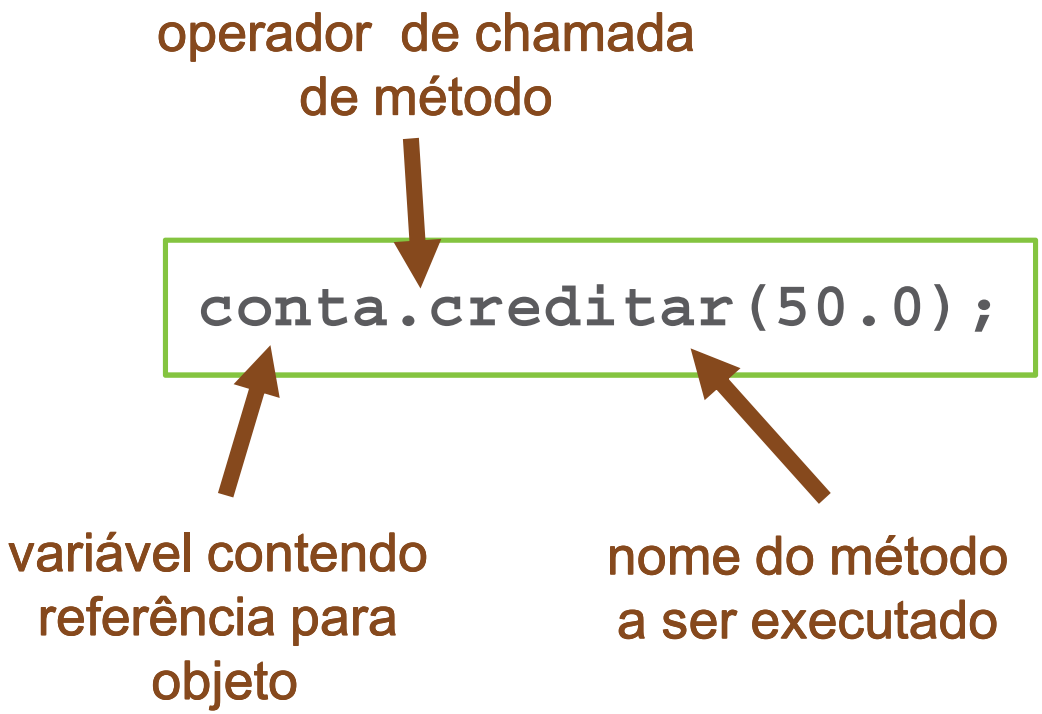
Exemplo (void)

```
class Conta{
      String numero;
      double saldo;

      void creditar(double valor) {
            saldo = saldo + valor;
      }

      ...
}
```

Métodos

- Por que no método creditar não temos o número da conta como parâmetro?
- Métodos são invocados por instâncias (objetos)
  - Também podem ser invocados pela classe (métodos estáticos)
- Parâmetros são passados por cópia

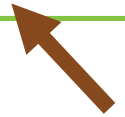
Chamada de Métodos



Instanciando Objetos

- Objetos precisam ser criados antes de serem utilizados
- A criação é feita com o operador **new**

```
Conta c = new Conta();
```



construtor

Construtores

- Constutores definem como os atributos do objeto vão ser inicializados
- São **semelhantes** a métodos, mas não têm tipo de retorno
- O nome do construtor deve ser **exatamente** o nome da classe
- Um classe pode ter diversos construtores, diferenciados pelos parâmetros

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

Construtor default

- Caso não seja definido um construtor, um construtor default é fornecido implicitamente
- O construtor default inicializa os atributos com seus valores padrões
- O construtor default não tem parâmetros
- Quando um construtor é definido, o construtor default não é mais gerado

```
public Conta() {
        ...
}
```

Valores padrões para atributos

Tipo	Valor Padrão
byte, short, int, long	0
float	0.0f
double	0.0
char	‘\u0000’
Tipos referência (String, arrays, objetos em geral)	null
boolean	false

Remoção de Objetos

- Em Java, não temos um método para remoção explícita de objetos da memória (como o free() de C++);
- Gargabe Collector (coletor de lixo) elimina objetos da memória quando eles não são mais referenciados;
- A JVM que decide a hora que será feita a coleta de lixo;

Modificadores

- Acesso
  - public
  - protected
  - private
  - default
- Outros
  - static
  - final
  - native
  - transient
  - synchronized

Modificadores de Acesso

- São aplicados a:
  - Classes;
  - Atributos;
  - Métodos ;
  - Construtores;
- Não se aplicam a variáveis locais;

public

- Classe: pode ser instanciada por qualquer outra classe
- Atributos: podem ser acessados por objetos de qualquer classe
- Métodos: podem ser chamados por métodos de qualquer classe

```
public class Conta{
      public String numero;
      ...
      public void debitar(double valor) {
            saldo = saldo - valor;
      }
      ...
}
```

protected

- Classe: não se aplica
- Atributos: podem ser acessados por objetos de classes dentro do mesmo pacote ou de qualquer subclasse da classe ao qual ele pertence
- Métodos: podem ser chamados por objetos de classes dentro do mesmo pacote ou de qualquer subclasse da classe ao qual ele pertence

```
public class Conta{
        protected String numero;
        ...
        protected void debitar(double valor) {
                saldo = saldo - valor;
        }
        ...
}
```

default

- Classe: visível apenas por classes do mesmo pacote
- Atributos: podem ser acessados por objetos de classes dentro do mesmo pacote
- Métodos: podem ser chamados por objetos de classes dentro do mesmo pacote

```
class Conta{
        String numero;
        ...
        void debitar(double valor) {
                saldo = saldo - valor;
        }
        ...
}
```

private

- Atributos: podem ser acessados apenas por objetos da mesma classe
- Métodos: podem ser chamados por objetos da mesma classe

```
class Conta{
        private String numero;
        ...
        private void debitar(double valor) {
                saldo = saldo - valor;
        }
        ...
}
```