# Programação orientada a objetos

Passagem e recebimento de objetos
Listas de objetos

# Passagem por parâmetro

- Todos já sabem que podemos passar por parâmetro nos métodos algumas varáveis/atributos.
- Mas o que é um objeto? Será que não podemos enviar um objeto?
- Sim , podemos. Quando você passa um objeto por parâmetro, na verdade você está passando uma referência para esse objeto.
- Isso significa que, quando você chama um método e passa um objeto como parâmetro, o método terá acesso a todas as propriedades e métodos desse objeto.

# Passagem de objetos por parâmetro

- Para isso, precisamos:
  - Defina o objeto que você deseja passar como parâmetro.
  - Defina o método que receberá o objeto como parâmetro.
  - Declare o parâmetro como o tipo do objeto que você está passando.

# Exemplo 1

```
3  public class Pessoa {
4      private String nome;
5      private int idade;
6
7      public Pessoa(String nome, int idade) {
8          this.nome = nome;
9          this.idade = idade;
10     }
11
12     public String getNome() {
13         return nome;
14     }
15
16     public int getIdade() {
17         return idade;
18     }
19 }
```

# Exemplo 1

```
3  public class Principal1 {
4      public static void main(String[] args) {
5          Pessoa pessoa = new Pessoa("João", 30);
6          meuMetodo(pessoa);
7      }
8
9      public static void meuMetodo(Pessoa p) {
10         System.out.println("Nome: " + p.getNome());
11         System.out.println("Idade: " + p.getIdade());
12     }
13 }
```

- Os métodos static ou métodos da classe são funções que não dependem de nenhuma variável de instância, quando invocados executam uma função sem a dependência do conteúdo de um objeto ou a execução da instância de uma classe, conseguindo chamar direto qualquer método da classe e também manipulando alguns campos da classe.

# Retornando objetos

- Também é possível retornar objetos em métodos.
- É muito útil para envio e recebimentos de novos objetos para serem trabalhados em outras classes.

# Exemplo 3

```
3  public class Produto {
4      private String nome;
5      private double preco;
6
7      public Produto(String nome, double preco) {
8          this.nome = nome;
9          this.preco = preco;
10     }
11
12     public String getNome() {
13         return nome;
14     }
15
16     public double getPreco() {
17         return preco;
18     }
19
20     public void setPreco(double preco) {
21         this.preco = preco;
22     }
23
24     public Produto clone() {
25         return new Produto(this.nome, this.preco);
26     }
27 }
```

```java
public class Produto {
    private String nome;
    private double preco;

    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }

    public String getNome() {
        return nome;
    }

    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    public Produto clone() {
        return new Produto(this.nome, this.preco);
    }
}
```

# Exemplo 3

```
3  public class Principal4 {
4      public static void main(String[] args) {
5              Produto produto1 = new Produto("Caneta", 1.5);
6              Produto produto2 = produto1.clone();
7
8              System.out.println("Produto 1 - Nome: " + produto1.getNome() + ", Preço: " + produto1.getPreco());
9              System.out.println("Produto 2 - Nome: " + produto2.getNome() + ", Preço: " + produto2.getPreco());
10
11             produto2.setPreco(2.0);
12
13             System.out.println("Produto 1 - Nome: " + produto1.getNome() + ", Preço: " + produto1.getPreco());
14             System.out.println("Produto 2 - Nome: " + produto2.getNome() + ", Preço: " + produto2.getPreco());
15         }
16 }
```

# Exemplo 3 V2

```
3  public class Produto {
4      private String nome;
5      private double preco;
6
7      public Produto(String nome, double preco) {
8          this.nome = nome;
9          this.preco = preco;
10     }
11
12     public double getPreco() {
13         return preco;
14     }
15
16     public void setPreco(double preco) {
17         this.preco = preco;
18     }
19
20     public String getNome() {
21         return nome;
22     }
23
24     public Produto clone() {
25         return this;
26     }
27 }
```

```java
public class Produto {
    private String nome;
    private double preco;

    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }

    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    public String getNome() {
        return nome;
    }

    public Produto clone() {
        return this;
    }
}
```

# Exemplo 3 V2

```
3  public class Principal1 {
4      public static void main (String[] args) {
5          Produto produto1 = new Produto("Caneta", 1.5);
6          Produto produto2 = produto1.clone();
7
8          System.out.println("Produto 1 - Nome: "+produto1.getNome()+", Preco "+produto1.getPreco());
9          System.out.println("Produto 2 - Nome: "+produto2.getNome()+", Preco "+produto2.getPreco());
10
11         produto2.setPreco(2.3);
12         System.out.println("Produto 1 - Nome: "+produto1.getNome()+", Preco "+produto1.getPreco());
13         System.out.println("Produto 2 - Nome: "+produto2.getNome()+", Preco "+produto2.getPreco());
14
15         if(produto1 == produto2)
16             System.out.println("São iguals.");
17         else
18             System.out.println("São diferentes");
19     }
20
21 }
```

# Relacionamento entre classes

- É possível termos objetos como atributos de classe, além de somente tipos primitivos (int, double, char)

# Exemplo 4

```
3  public class Endereco {
4      private String rua;
5      private int numero;
6
7      public Endereco(String rua, int numero) {
8          this.rua = rua;
9          this.numero = numero;
10     }
11
12     public String getRua() {
13         return rua;
14     }
15
16     public int getNumero() {
17         return numero;
18     }
19 }
```

# Exemplo 4

```
3  public class Pessoa2 {
4      private String nome;
5      private int idade;
6      private Endereco endereco;
7
8      public Pessoa2(String nome, int idade, Endereco endereco) {
9          this.nome = nome;
10         this.idade = idade;
11         this.endereco = endereco;
12     }
13
14     public String getNome() {
15         return nome;
16     }
17
18     public int getIdade() {
19         return idade;
20     }
21
22     public Endereco getEndereco() {
23         return endereco;
24     }
25 }
```

# Exemplo 4

```
3  public class Principal3 {
4      public static void main(String[] args) {
5          Endereco endereco = new Endereco("Rua 1", 123);
6          Pessoa2 pessoa = new Pessoa2("João", 30, endereco);
7
8          System.out.println("Nome: " + pessoa.getNome());
9          System.out.println("Idade: " + pessoa.getIdade());
10         System.out.println("Endereço: " + pessoa.getEndereco().getRua() + ", "
11                 + "" + pessoa.getEndereco().getNumero());
12     }
13 }
```

# Exemplo 4 – V2

```
3  public class Principal3 {
4      public static void main(String[] args) {
5
6          Pessoa2 pessoa = new Pessoa2("João", 30, new Endereco("Rua 1", 123));
7
8          System.out.println("Nome: " + pessoa.getNome());
9          System.out.println("Idade: " + pessoa.getIdade());
10         System.out.println("Endereço: " + pessoa.getEndereco().getRua() + ", "
11                 + "" + pessoa.getEndereco().getNumero());
12
13     }
14 }
```

# Listas

- As listas em Java são estruturas de dados que permitem armazenar e manipular coleções de objetos. Elas são implementadas pela interface *List* e podem ser criadas usando classes como *ArrayList*, *LinkedList* e *Vector*.

- A interface *List* é uma subinterface da interface *Collection* e define um contrato para classes que implementam listas em Java. Ela estende a interface *Iterable*, o que significa que as listas podem ser percorridas usando um loop for-each.

- As listas em Java podem conter objetos de qualquer tipo, incluindo tipos primitivos, como int, double e boolean.

# Exemplo 5 – Mesma classe do exemplo 1

```
3  public class Pessoa {
4      private String nome;
5      private int idade;
6
7      public Pessoa(String nome, int idade) {
8          this.nome = nome;
9          this.idade = idade;
10     }
11
12     public String getNome() {
13         return nome;
14     }
15
16     public int getIdade() {
17         return idade;
18     }
19 }
```

# Exemplo 5

```java
import java.util.ArrayList;
import java.util.List;

public class Principal5 {
    public static void main(String[] args) {
        List<Pessoa> listaPessoas = new ArrayList<Pessoa>();

        Pessoa pessoa1 = new Pessoa("João", 30);
        Pessoa pessoa2 = new Pessoa("Maria", 25);
        Pessoa pessoa3 = new Pessoa("Pedro", 40);

        listaPessoas.add(pessoa1);
        listaPessoas.add(pessoa2);
        listaPessoas.add(pessoa3);

        // Chamando um método e passando a lista como parâmetro
        exibirPessoas(listaPessoas);
    }

    public static void exibirPessoas(List<Pessoa> lista) {
        for (Pessoa p : lista) {
            System.out.println("Nome: " + p.getNome());
            System.out.println("Idade: " + p.getIdade());
        }
    }
}
```

```java
import java.util.ArrayList;
import java.util.List;

public class Principal5 {
    public static void main(String[] args) {
        List<Pessoa> listaPessoas = new ArrayList<Pessoa>();

        Pessoa pessoa1 = new Pessoa("João", 30);
        Pessoa pessoa2 = new Pessoa("Maria", 25);
        Pessoa pessoa3 = new Pessoa("Pedro", 40);

        listaPessoas.add(pessoa1);
        listaPessoas.add(pessoa2);
        listaPessoas.add(pessoa3);

        // Chamando um método e passando a lista como parâmetro
        exibirPessoas(listaPessoas);
    }

    public static void exibirPessoas(List<Pessoa> lista) {
        for (Pessoa p : lista) {
            System.out.println("Nome: " + p.getNome());
            System.out.println("Idade: " + p.getIdade());
        }
    }
}
```

# Listas

```
26      private static void exibirPessoas2(List<Pessoa> lista) {
27          for(int i=0;i<lista.size();i++) {
28              System.out.println("Nome: "+lista.get(i).getNome());
29              System.out.println("Idade: " + lista.get(i).getIdade());
30          }
31      }
```

# Listas

- A interface List define vários métodos que permitem manipular os elementos da lista. Aqui estão alguns dos métodos mais comuns:
  - add(Object elemento): adiciona um elemento à lista.
  - add(int indice, Object elemento): adiciona um elemento à lista em um índice específico.
  - remove(Object elemento): remove um elemento da lista.
  - remove(int indice): remove um elemento da lista em um índice específico.
  - get(int indice): retorna o elemento da lista no índice especificado.
  - set(int indice, Object elemento): substitui o elemento na lista no índice especificado pelo elemento especificado.
  - size(): retorna o número de elementos na lista.

# Jogo rápido

- Teste os métodos do slide anterior no exemplo 5.

# Exercícios - Relacionamento

- 1 - Crie uma classe Aluno com os atributos nome e notaFinal. Em seguida, crie uma classe Boletim com um método imprimirStatus(Aluno a) que imprime se o aluno foi aprovado (nota ≥ 6) ou reprovado.

- 2 - Crie uma classe Produto com os atributos nome e preco. Em outra classe, crie um método criarProdutoDesconto(String nome, double preco) que retorna um objeto Produto com 10% de desconto aplicado ao preço.

- 3 - Crie uma classe Livro com os atributos titulo e autor. Em seguida, crie uma lista de livros (ArrayList<Livro>) e um método que recebe a lista e imprime os dados de cada livro.

- 4 - Crie uma classe Conta com os atributos titular e saldo. Crie um método depositar(Conta c, double valor) que receba a conta como parâmetro e aumente o saldo.

# Exercícios - Listas

- 1 - Crie uma classe Produto com atributos nome, preco e quantidade. Crie uma lista de produtos e adicione alguns produtos nessa lista. Em seguida, percorra a lista e imprima os dados de cada produto.

- 2 - Crie uma classe Aluno com atributos nome, nota1 e nota2. Crie uma lista de alunos e adicione alguns alunos nessa lista. Em seguida, percorra a lista e calcule a média de cada aluno. Se a média for maior ou igual a 6, imprima que o aluno foi aprovado. Caso contrário, imprima que o aluno foi reprovado.

- 3 - Crie uma classe Pessoa com atributos nome, idade e sexo. Crie uma lista de pessoas e adicione algumas pessoas nessa lista. Em seguida, crie um método que recebe uma lista de pessoas e retorna a quantidade de mulheres. Por fim, chame esse método passando a lista de pessoas e imprima a quantidade de mulheres.

# Exercícios

- 4 - Crie uma classe Livro com atributos titulo, autor e ano. Crie uma lista de livros e adicione alguns livros nessa lista. Em seguida, ordene a lista de livros pelo ano de lançamento e imprima os dados de cada livro.

- 5 - Crie uma classe Conta com atributos numero, titular e saldo. Crie uma lista de contas e adicione algumas contas nessa lista. Em seguida, crie um método que recebe uma lista de contas e retorna a conta com o maior saldo. Por fim, chame esse método passando a lista de contas e imprima os dados da conta com o maior saldo.