Apostila de Exercícios

JSE – F3

Fundamentos de Programação Avançado

Exercício 1

```
package exer1;

import static java.lang.System.exit;
import static java.lang.System.out;
import static javax.swing.JOptionPane.showMessageDialog;

import javax.swing.JOptionPane;

public class Exer1 {
  public static void main(String[] args) {
    System.out.println("Ola 1");
    out.println("Ola 2");

    JOptionPane.showMessageDialog(null, "Ola 3");
    showMessageDialog(null, "Ola 4");

    System.exit(0);
    exit(0);
  }
}
```

Exercício 2

```java
package exer2;

public class Exer2 {

  public static long somar(int v1, int v2) {
    return v1 + v2;
  }

  public static void main(String[] args) {
    System.out.println(somar(10, 10));
    // System.out.println(somar(10, 10, 10)); Não tem como fazer....

    System.out.println(adicionar(10, 10));
    System.out.println(adicionar(10, 10, 1));
    System.out.println(adicionar(10, 10, 1, 2));
    System.out.println(adicionar(10, 10, 1, 2, 3));
    System.out.println(adicionar(10, 10, 1, 2, 3, 4));
  }

  public static long adicionar(int... parans) {
    long res = 0;
    for (int i : parans) {
      res += i;
    }
    return res;
  }
}
```

Exercício 3

```
package exer3;

public class Exer3 {

  public static void main(String[] args) {
    // Java 1.4
    Integer v1 = new Integer(10);
    int dez = v1.intValue();
    System.out.println(dez);
    int soma  = 1 + v1.intValue() + 3;
    System.out.println(soma);

    // Java 5.0
    Integer o1 = 10; // autobox
    int v2 = o1; //unbox

    int total = 10 + o1 + v2; // unbox e autobox
    System.out.println(total);

    Double x = 1.50;
    for (int i = 0; i < 10; i++) {
      x++;
    }
    System.out.println(x);
  }
}
```

Exercício 4

```java
package exer4;

public class Exer4 {
  public static void main(String[] args) {
    StringBuilder str = new StringBuilder();

    // Adicionando conteúdo
    str.append("Fernando");
    str.append(" Franzini");

    // obter conteúdo completo
    System.out.println(str.toString());

    // obter parte do conteúdo
    System.out.println(str.substring(4, 10));

    // verificar tamanho
    System.out.println(str.length());

    // deleter uma parte
    str = str.delete(8, str.length());
    System.out.println(str.toString());

    // deletar tudo
    str.delete(0, str.length());
    System.out.println("str=" + str.toString());
  }
}
```

Exercício 5

```java
package exer5;

import java.io.BufferedReader;

public class Exer5 {
  public static void main(String[] args) {
    // 1. Mostrar o compilador obrigando a tratar.
    // 2. Fazer o try manual.
    // 3. Ensinar o eclipse a gerar o try automático.
    // 4. Depurar uma leitura com erro e mostrar o printStackTrace();

    FileReader arquivo = null;
    try {
      arquivo = new FileReader("D:/fernando.txt");
    } catch (FileNotFoundException e) {
      e.printStackTrace();
      System.exit(0);
    }
    BufferedReader leitor = new BufferedReader(arquivo);
    try {
      String conteudo = leitor.readLine();
      JOptionPane.showMessageDialog(null, "Conteudo do arquivo:\n" + conteudo);
    } catch (IOException e) {
      e.printStackTrace();
    }
    try {
      leitor.close();
      arquivo.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
    System.exit(0);
  }
}
```

Exercício 6

```
package exer6;

import java.io.File;

public class Exer6 {

  public static void main(String[] args) {
    // 1. Fazer e executar com try
    String nome = null;
    try {
      File arquivo = new File(nome);
      System.out.println(arquivo);
    } catch (NullPointerException e) {
      // e.printStackTrace();
      System.out.println("não abriu o arquivo");
    }
    System.out.println("continua o programa...");
    try {
      int res = 10 / 0;
    } catch (Exception e) {
      // e.printStackTrace();
      System.out.println("não fez a operação");
    }
    System.out.println("continua o programa...");
  }

  public static void main2(String[] args) {
    // 2. Fazer e executar sem try
    String nome = null;
    File arquivo = new File(nome);
    System.out.println(arquivo);
    System.out.println("continua o programa...");
    int res = 10 / 0;
    System.out.println("continua o programa...");
  }
}
```

Exercício 7

```
package exer7;

public class Pessoa {
  public int calcularIdade(Integer anoNasc, Integer anoAtual) throws Exception {
    if (anoNasc == null) {
      throw new Exception("Ano nascimento é obrigatório.");
    }
    if (anoAtual == null) {
      throw new Exception("Ano atual é obrigatório.");
    }
    if (anoAtual <= anoNasc) {
      throw new Exception("Ano atual deve ser maior que data nasc.");
    }
    return anoAtual - anoNasc;
  }
}
```

```
public class Exer7 {
  public static void main(String[] args) {
    Pessoa joao = new Pessoa();
    // 1. exemplo
    try {
      joao.calcularIdade(null, null);
    } catch (Exception e) {
      e.printStackTrace();
    }
    // 2. exemplo
    try {
      joao.calcularIdade(0, null);
    } catch (Exception e) {
      e.printStackTrace();
    }
    // 3. exemplo
    try {
      joao.calcularIdade(0, 0);
    } catch (Exception e) {
      e.printStackTrace();
    }
    // 4. exemplo
    try {
      joao.calcularIdade(1980, 1979);
    } catch (Exception e) {
      e.printStackTrace();
    }
    // 5. exemplo
    try {
      System.out.println("idade é = " + joao.calcularIdade(1980, 1981));
      System.out.println("idade é = " + joao.calcularIdade(1980, 2016));
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Exercício 8

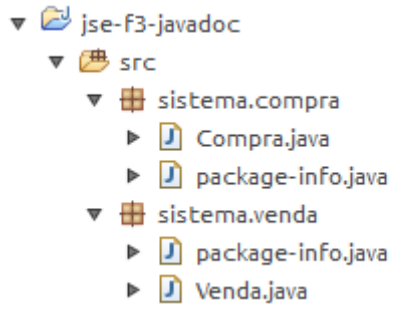
```java
package exer8;

public class ObrigatorioException extends Exception {
  public ObrigatorioException(String campo) {
    super("Campo obrigatório - " + campo + ".");
  }
}
```

```java
package exer8;

public class InvalidoException extends Exception {
  public InvalidoException(String mensagem) {
    super("Operação Inválida - " + mensagem + ".");
  }
}
```

```java
package exer8;

public class Pessoa {
  public int calcularIdade(Integer anoNasc, Integer anoAtual)
      throws ObrigatorioException, InvalidoException {
    if (anoNasc == null) {
      throw new ObrigatorioException("Ano nasc");
    }
    if (anoAtual == null) {
      throw new ObrigatorioException("Ano atual");
    }
    if (anoAtual <= anoNasc) {
      throw new InvalidoException("Ano atual deve ser maior que data nasc");
    }
    return anoAtual - anoNasc;
  }
}
```

```
public class Exer8 {
  public static void main(String[] args) {
    Pessoa joao = new Pessoa();
    // 1. exemplo
    try {
      joao.calcularIdade(null, null);
    } catch (ObrigatorioException | InvalidoException e) {
      e.printStackTrace();
    }
    // 2. exemplo
    try {
      joao.calcularIdade(0, null);
    } catch (ObrigatorioException | InvalidoException e) {
      e.printStackTrace();
    }
    // 3. exemplo
    try {
      joao.calcularIdade(0, 0);
    } catch (ObrigatorioException | InvalidoException e) {
      e.printStackTrace();
    }
    // 4. exemplo
    try {
      joao.calcularIdade(1980, 1979);
    } catch (ObrigatorioException | InvalidoException e) {
      e.printStackTrace();
    }
    // 5. exemplo
    try {
      System.out.println("idade é = " + joao.calcularIdade(1980, 2016));
    } catch (ObrigatorioException | InvalidoException e) {
      e.printStackTrace();
    }
  }
}
```

Exercício 9



package-info.java

```
/**
 * Providência classes utilizadas no processo de compras.
 * @author Fernando Franzini
 */
package sistema.compra;
```

package-info.java

```
/**
 * Providência classes utilizadas no processo de vendas.
 * @author Fernando Franzini
 */
package sistema.venda;
```

```java
/*
 * Todos os direito reservados a empresa XYZ.
 */
package sistema.compra;

/**
 * Representa uma operação de <b>compra</b>. Em caso de dúvida, veja link
 * <a href="http://fernandofranzini.wordpress.com">tutorial</a>.
 * @author Fernando Franzini
 * @version 1.0 - 13/09/16
 */
public class Compra {

  /** Armazena o valor total. **/
  private double valor;

  /**
   * Constroe uma nova compra com valores padrões.
   */
  public Compra() {
    valor = 10;
  }

  /**
   * Processa uma nova compra.
   * @param valor valor da operação de compra.
   * @throws Exception caso ocorrer erros na regras de compra.
   */
  public void compra(double valor) throws Exception {
  }

  /**
   * Retorna o valor da compra.
   * @return valor da compra.
   */
  public double getValor() {
    return valor;
  }

  /**
   * Informa um novo valor da compra.
   * @param valor novo valor a ser informado.
   */
  public void setValor(double valor) {
    this.valor = valor;
  }
}
```

```java
/*
 * Todos os direito reservados a empresa XYZ.
 */
package sistema.venda;

/**
 * Representa uma operação de <b>venda</b>.
 * @author Fernando Franzini
 * @version 1.0 - 13/09/16
 */
public class Venda {

  /** Armazena o valor total. **/
  private double valor;

  /**
   * Cria uma nova compra sem valor padrão.
   */
  public Venda() {
  }

  /**
   * Processa uma nova venda.
   * @param valor valor da operação de venda.
   * @throws Exception caso ocorrer erros na regras de venda.
   */
  public void vender(double valor) throws Exception {
  }

  /**
   * Retorna o valor da venda.
   * @return valor da venda.
   */
  public double getValor() {
    return valor;
  }

  /**
   * Informa um novo valor da venda.
   * @param valor novo valor a ser informado.
   */
  public void setValor(double valor) {
    this.valor = valor;
  }
}
```

- Gerar Java Doc.

Exercício 10

```java
package exer10;

public interface Cantor {
  void cantar();
}
```

```java
package exer10;

public interface Jogador {
  void jogar(String esporte);
}
```

```java
package exer10;

// Implementar 1 de cada vez, explicando o contrato.
// Explicar atalho do eclipse para geração do contrato.
public class Pessoa implements Cantor, Jogador {

  @Override
  public void cantar() {
  }

  @Override
  public void jogar(String esporte) {
  }
}
```

Exercício 11

```java
package exer11;

public class Torcedor {
  public void torcer() {
    System.out.println("Torcendo para meu time.");
  }
}
```

```java
package exer11;

public class Corinthians extends Torcedor {
  public void torcer() {
    System.out.println("Timão eooo.");
  }
}
```

```java
package exer11;

public class Palmeiras extends Torcedor {
  public void torcer() {
    System.out.println("porcooooooo.");
  }
}
```

```java
package exer11;

public class Santos extends Torcedor {
}
```

```
package exer11;

public class Exer11 {
  public static void main(String[] args) {
    // 1. Chamada do metodo normal.
    Torcedor torcedor = new Torcedor();
    torcedor.torcer();

    // 2. Chamadas polimorficas
    torcedor = new Corinthians();
    torcedor.torcer();

    torcedor = new Palmeiras();
    torcedor.torcer();

    torcedor = new Santos();
    torcedor.torcer();
  }
}
```

Exercício 12

```java
package exer12;

public interface Torcedor {
   void tocer();
}
```

```java
package exer12;

public class Palmeiras implements Torcedor {
   @Override
   public void tocer() {
      System.out.println("Porcoo eooo");
   }
}
```

```java
package exer12;

public class Corinthians implements Torcedor {
   @Override
   public void tocer() {
      System.out.println("Timão eoooo");
   }
}
```

```java
package exer12;

public class Exer12 {
   public static void main(String[] args) {
      Torcedor t = new Corinthians();
      t.tocer();
      t = new Palmeiras();
      t.tocer();
   }
}
```

Exercício 13

```
package exer13;

public class PizzaCalabresa {
  public void preparar() {
    System.out.println("molho, queijo, calabreza, cebola e tomate.");
  }

  public void assar() {
    System.out.println("15  minutos.");
  }

  public void cobrar() {
    System.out.println("R$ 12,00");
  }
}
```

```
package exer13;

public class Forno {
  public void fabricar(PizzaCalabresa calabreza) {
    calabreza.preparar();
    calabreza.assar();
    calabreza.cobrar();
  }
}
```

```
package exer13;

public class Programa {
  public static void main(String[] args) {
    Forno forno = new Forno();
    PizzaCalabresa calabresa = new PizzaCalabresa();
    forno.fabricar(calabresa);
  }
}
```

Exercício 14

```
package exer14;

public class PizzaCalabresa {
  public void preparar() {
    System.out.println("molho, queijo, calabreza, cebola e tomate.");
  }

  public void assar() {
    System.out.println("15  minutos.");
  }

  public void cobrar() {
    System.out.println("R$ 12,00");
  }
}
```

```
package exer14;

public class PizzaNapolitana {
  public void preparacao() {
    System.out.println("molho, presunto, queijo, tomate e oregano.");
  }

  public void assamento() {
    System.out.println("19  minutos.");
  }

  public void valor() {
    System.out.println("R$ 18,00");
  }
}
```

```
package exer14;

public class Programa {
  public static void main(String[] args) {
    Forno forno = new Forno();

    PizzaCalabresa calabresa = new PizzaCalabresa();
    forno.fabricar(calabresa);

    PizzaNapolitana napolitana = new PizzaNapolitana();
    // Deixe o erro de compilação acontencer.
    forno.fabricar(napolitana);
  }
}
```

```java
package exer14;

public class Forno {
  // 1 PARTE: Execute o programa tentando passar a napolitana no método da
  // calabreza que é parte do aprendizado.
  public void fabricar(PizzaCalabresa calabreza) {
    calabreza.preparar();
    calabreza.assar();
    calabreza.cobrar();
  }

  // 2 PARTE: Crie o novo método para fabricar napolitana e
  // conclua a explicação.
  public void fabricar(PizzaNapolitana napolitana) {
    napolitana.preparacao();
    napolitana.assamento();
    napolitana.valor();
  }
}
```

Exercício 15

```
package exer15;

public interface Pizza {
  void preparar();
   void assar();
  void cobrar();
}
```

```
package exer15;

public class Forno {

  public void fabricar(Pizza pizza) {
    // pizza é uma referencia polimórfica = variável e dinâmica.
    pizza.preparar();
    pizza.assar();
    pizza.cobrar();
  }
}
```

```
package exer15;

public class Calabresa implements Pizza {
  @Override
  public void preparar() {
    System.out.println("molho, queijo, calabreza, cebola e tomate.");
  }

  @Override
  public void assar() {
    System.out.println("15  minutos.");
  }

  @Override
  public void cobrar() {
    System.out.println("R$ 12,00");
  }
}
```

```java
package exer15;

public class Napolitana implements Pizza {
  @Override
  public void preparar() {
    System.out.println("molho, presunto, queijo, tomate e oregano.");
  }

  @Override
  public void assar() {
    System.out.println("19  minutos.");
  }

  @Override
  public void cobrar() {
    System.out.println("R$ 18,00");
  }
}
```

```java
package exer15;

public class Programa {
  public static void main(String[] args) {
    Forno forno = new Forno();

    Calabresa calabresa = new Calabresa();
    Napolitana napolitana = new Napolitana();

    forno.fabricar(calabresa);
    forno.fabricar(napolitana);
  }
}
```

Exercício 16

```
package exer16;

public interface Pizza {
   void preparar();
    void assar();
   void cobrar();
}
```

```
package exer16;

public class Calabresa implements Pizza {
   @Override
   public void preparar() {
      System.out.println("molho, queijo, calabreza, cebola e tomate.");
   }

   @Override
   public void assar() {
      System.out.println("15  minutos.");
   }

   @Override
   public void cobrar() {
      System.out.println("R$ 12,00");
   }
}
```

```
package exer16;

public class Napolitana implements Pizza {
    @Override
    public void preparar() {
      System.out.println("molho, presunto, queijo, tomate e oregano.");
    }

    @Override
    public void assar() {
      System.out.println("19  minutos.");
    }

    @Override
    public void cobrar() {
      System.out.println("R$ 18,00");
    }
}
```

```java
package exer16;

public class AlhoBacon implements Pizza {
  @Override
  public void preparar() {
    System.out.println("molho, alho e muito bacon.");
  }

  @Override
  public void assar() {
    System.out.println("14  minutos.");
  }

  @Override
  public void cobrar() {
    System.out.println("R$ 15,50");
  }
}
```

```java
package exer16;

public class Forno {
  public void fabricar(Pizza pizza) {
    // Uma única alteração no forno, altera todas as pizzas.
    System.out.println("Promoção: borda de catupiri.");
    pizza.preparar();
    pizza.assar();
    pizza.cobrar();
  }
}
```

```java
package exer16;

public class Programa {
  public static void main(String[] args) {
    Forno forno = new Forno();

    Calabresa calabresa = new Calabresa();
    Napolitana napolitana = new Napolitana();
    AlhoBacon alhoBacon = new AlhoBacon();

    forno.fabricar(calabresa);
    forno.fabricar(napolitana);
    forno.fabricar(alhoBacon);
  }
}
```

Exercício 17

```java
package exer17;

public interface TipoPagamento {
  void pagar(double valor);
}
```

```java
package exer17;

public class Boleto implements TipoPagamento {
  @Override
  public void pagar(double valor) {
    System.out.println("Pagamento em boleto. Valor da taxa fixa R$ 0.80 centavos.");
  }
}
```

```java
package exer17;

public class CartaoCredito implements TipoPagamento {
  @Override
  public void pagar(double valor) {
    double taxa = valor * 0.2;
    System.out.println("Pagamento em cartão. Valor da taxa é R$ " + taxa);
  }
}
```

```java
package exer17;

public class Dinheiro implements TipoPagamento {
  @Override
  public void pagar(double valor) {
    System.out.println("Pagamento em dinheiro. Não tem taxa!");
  }
}
```

```java
package exer17;

public class Venda {
  public void vender(TipoPagamento pgto, double valor) {
    pgto.pagar(valor);
  }
}
```

```java
package exer17;

public class Exer17 {
  public static void main(String[] args) {
    Venda venda = new Venda();
    venda.vender(new Dinheiro(), 100.00);
    venda.vender(new CartaoCredito(), 100.00);
    venda.vender(new Boleto(), 100.00);
  }
}
```

Exercício 18

```java
package exer18;

public interface Entrega {
  void entregar(String produto);
}
```

```java
package exer18;

public class Pac implements Entrega {
  @Override
  public void entregar(String produto) {
    System.out.println("Entrega via pac = " + produto + ", 15 dias de prazo.");
  }
}
```

```java
package exer18;

public class Sedex implements Entrega {
  @Override
  public void entregar(String produto) {
    System.out.println("Entrega via sedex = " + produto + ", 24 horas.");
  }
}
```

```java
package exer18;

public class Compra {
  private Entrega entrega;

  public Compra(Entrega e) {
    entrega = e;
  }

  public void comprar(String produto) {
    entrega.entregar(produto);
  }
}
```

```java
package exer18;

public class Exer18 {
  public static void main(String[] args) {
    Compra dvd = new Compra(new Pac());
    dvd.comprar("Hobbit 1");

    Compra livro = new Compra(new Sedex());
    livro.comprar("Use a cabeça Java");
  }
}
```

Exercício 19

```java
package exer19;

import java.util.ArrayList;

public class Exer19 {

  public static void main(String[] args) {
    ArrayList<String> colecao = new ArrayList<>();
    // List<String> colecao = new ArrayList<>(); //ou polimorfico

    // verificando tamanho
    System.out.println(colecao.size());

    // Adicionando
    colecao.add("Fernando");
    colecao.add("Rebeca");
    colecao.add("Anny");
    colecao.add("Jonas");
    // verificando tamanho
    System.out.println(colecao.size());

    // Iterando nos conteudo, ordenado pela inserção
    System.out.println("--> iteração 1");
    for (String item : colecao) {
      System.out.println(item);
    }

    // permite duplicado
    colecao.add("Fernando");

    // Iterando nos conteudo, ordenado pela inserção
    System.out.println("--> iteração 2");
    for (String item : colecao) {
      System.out.println(item);
    }

    // procurando um item
    System.out.println(colecao.contains("Rebeca"));
    System.out.println(colecao.contains("Neymar"));
```

```
        // remove itens
        System.out.println(colecao.remove("Rebeca"));
        System.out.println(colecao.remove("Adriano"));

        // Iterando nos conteudo, ordenado pela inserção
        System.out.println("--> iteração 3");
        for (String item : colecao) {
          System.out.println(item);
        }

        // Acessar por indice
        System.out.println("indice 1=" + colecao.get(1));
        System.out.println("indice 2=" + colecao.get(2));

        // limpar tudo
        colecao.clear();
        System.out.println(colecao.size());
    }
}
```

Exercício 20

```java
package exer20;

import java.util.HashSet;

public class Exer20 {

  public static void main(String[] args) {
    HashSet<String> colecao = new HashSet<>();
    //Set<String> colecao = new HashSet<>(); //ou polimorfico

    // verificando tamanho
    System.out.println(colecao.size());

    // Adicionando
    colecao.add("Fernando");
    colecao.add("Rebeca");
    colecao.add("Anny");
    colecao.add("Jonas");
    colecao.add("Fabio");
    colecao.add("Junior");

    // verificando tamanho
    System.out.println(colecao.size());

    // Iterando nos conteudo, sem regra de ordenção, pode variar de OS e JDK.
    System.out.println("--> iteração 1");
    for (String item : colecao) {
      System.out.println(item);
    }

    // Não permite duplicado
    System.out.println(colecao.add("Fernando"));

    // Iterando nos conteudo, sem regra de ordenção, pode variar de OS e JDK.
    System.out.println("--> iteração 2");
    for (String item : colecao) {
      System.out.println(item);
    }
```

```
        // procurando um item
        System.out.println(colecao.contains("Rebeca"));
        System.out.println(colecao.contains("Neymar"));

        // remove itens
        System.out.println(colecao.remove("Rebeca"));
        System.out.println(colecao.remove("Adriano"));

        // Iterando nos conteudo, sem regra de ordenção, pode variar de OS e JDK.
        System.out.println("--> iteração 3");
        for (String item : colecao) {
          System.out.println(item);
        }

        // limpar tudo
        colecao.clear();
        System.out.println(colecao.size());
    }
}
```

Exercício 21

```
package exer21;

import java.util.HashSet;

public class Exer21 {

  public static void main(String[] args) {
    //TreeSet<String> colecao = new TreeSet<>();
    Set<String> colecao = new HashSet<>(); //ou polimorfico

    // verificando tamanho
    System.out.println(colecao.size());

    // Adicionando
    colecao.add("Fernando");
    colecao.add("Rebeca");
    colecao.add("Anny");
    colecao.add("Jonas");
    colecao.add("Fabio");
    colecao.add("Junior");

    // verificando tamanho
    System.out.println(colecao.size());

    // Iterando nos conteudo, usa regra de ordenação por ordem alf em string.
    System.out.println("--> iteração 1");
    for (String item : colecao) {
      System.out.println(item);
    }

    // Não permite duplicado
    System.out.println(colecao.add("Fernando"));

    // Iterando nos conteudo, usa regra de ordenação por ordem alf em string.
    System.out.println("--> iteração 2");
    for (String item : colecao) {
      System.out.println(item);
    }
```

```
        // procurando um item
        System.out.println(colecao.contains("Rebeca"));
        System.out.println(colecao.contains("Neymar"));

        // remove itens
        System.out.println(colecao.remove("Rebeca"));
        System.out.println(colecao.remove("Adriano"));

        // Iterando nos conteudo, usa regra de ordenação por ordem alf em string.
        System.out.println("--> iteração 3");
        for (String item : colecao) {
          System.out.println(item);
        }

        // limpar tudo
        colecao.clear();
        System.out.println(colecao.size());
    }
}
```

Exercício 22

```java
package exer22;

import java.util.HashMap;

public class Exer22 {

  public static void main(String[] args) {
    HashMap<String, String> mapa = new HashMap<>();
    // Map<String, String> mapa = new HashMap<>(); // polimorfico

    // Adicionar chave
    mapa.put("tia", "Michele Ferreira");
    mapa.put("pai", "Fernando Franzini");
    mapa.put("mae", "Anny Carla");
    mapa.put("filha", "Rebeca Galdiano");
    mapa.put("cachorro", "Spot");

    // verificando tamanho
    System.out.println(mapa.size());

    // Buscando objetos por sua chave
    System.out.println(mapa.get("filha"));
    System.out.println(mapa.get("primo"));

    // Mapa não foi criado para ser iterado, e sim acessados pela chave.
    // mas a api disponibiliza .values() que converte o mapa em uma List.
    // Iterando nos conteudo, sem regra de ordenção, pode variar de OS e JDK.
    System.out.println("--> iteração 1");
    for (String item : mapa.values()) {
      System.out.println(item);
    }

    // Não permite chaves duplicação, é sobreposto.
    mapa.put("cachorro", "fofão");

    System.out.println("--> iteração 2");
    for (String item : mapa.values()) {
      System.out.println(item);
    }
```

```
        // Remover chaves
        mapa.remove("cachorro");

        System.out.println("--> iteração 3");
        for (String item : mapa.values()) {
          System.out.println(item);
        }

        // iterando via chaves, que retorna as chaves existentes.
        System.out.println("--> iteração 4");
        for (String item : mapa.keySet()) {
          System.out.println("chave=" + item + ", objeto=" + mapa.get(item));
        }

        // Verificar se contem a chave
        System.out.println(mapa.containsKey("filha"));
        System.out.println(mapa.containsKey("primo"));

        // Verificar se contem o objeto
        System.out.println(mapa.containsValue("Fernando Franzini"));
        System.out.println(mapa.containsKey("Ricardo Silva"));

        // limpar tudo
        mapa.clear();
        System.out.println(mapa.size());
    }
}
```

Exercício 23

```java
package exer23;

import java.util.LinkedHashMap;

public class Exer23 {

  public static void main(String[] args) {
    LinkedHashMap<String, String> mapa = new LinkedHashMap<>();
    //Map<String, String> mapa = new LinkedHashMap<>(); // polimorfico

    // Adicionar chave
    mapa.put("tia", "Michele Ferreira");
    mapa.put("pai", "Fernando Franzini");
    mapa.put("mae", "Anny Carla");
    mapa.put("filha", "Rebeca Galdiano");
    mapa.put("cachorro", "Spot");


    // verificando tamanho
    System.out.println(mapa.size());

    // Buscando objetos por sua chave
    System.out.println(mapa.get("filha"));
    System.out.println(mapa.get("primo"));

    // Mapa não foi criado para ser iterado, e sim acessados pela chave.
    // mas a api disponibiliza .values() que converte o mapa em uma List.
    // Iterando nos conteudo, mantem a ordem de inserção..
    System.out.println("--> iteração 1");
    for (String item : mapa.values()) {
      System.out.println(item);
    }

    // Não permite chaves duplicação, é sobreposto.
    mapa.put("cachorro", "fofão");

    System.out.println("--> iteração 2");
    for (String item : mapa.values()) {
      System.out.println(item);
    }
```

```
        // Remover chaves
        mapa.remove("cachorro");

        System.out.println("--> iteração 3");
        for (String item : mapa.values()) {
          System.out.println(item);
        }

        // iterando via chaves, que retorna as chaves existentes.
        System.out.println("--> iteração 4");
        for (String item : mapa.keySet()) {
          System.out.println("chave=" + item + ", objeto=" + mapa.get(item));
        }

        // Verificar se contem a chave
        System.out.println(mapa.containsKey("filha"));
        System.out.println(mapa.containsKey("primo"));

        // Verificar se contem o objeto
        System.out.println(mapa.containsValue("Fernando Franzini"));
        System.out.println(mapa.containsKey("Ricardo Silva"));

        // limpar tudo
        mapa.clear();
        System.out.println(mapa.size());
    }
}
```

Exercício 24

```java
package exer24;

import java.util.LinkedList;

public class Exer24 {

  public static void main(String[] args) {
    LinkedList<String> fila = new LinkedList<>();
    // Queue<String> fila = new LinkedList<>(); // ou polimorfico

    // verificando tamanho
    System.out.println(fila.size());

    // Adicionando
    fila.add("Fernando");
    fila.add("Rebeca");
    fila.add("Anny");
    fila.add("Jonas");

    // verificando tamanho
    System.out.println(fila.size());

    // Iterando nos conteudo, ordenado pela inserção
    System.out.println("--> iteração 1");
    for (String item : fila) {
      System.out.println(item);
    }

    // permite duplicado
    fila.add("Fernando");

    // Iterando nos conteudo, ordenado pela inserção
    System.out.println("--> iteração 2");
    for (String item : fila) {
      System.out.println(item);
    }

    // procurando um item
    System.out.println(fila.contains("Rebeca"));
    System.out.println(fila.contains("Neymar"));
```

```
        // remove itens
        System.out.println(fila.remove("Rebeca"));
        System.out.println(fila.remove("Adriano"));

        // Iterando nos conteudo, ordenado pela inserção
        System.out.println("--> iteração 3");
        for (String item : fila) {
            System.out.println(item);
        }

        // Processando a fila, pega o elemento e ja remove.
        System.out.println("--> Processando a fila");
        String item = null;
        while ((item = fila.poll()) != null) {
            System.out.println("fila=" + item);
        }
        System.out.println(fila.size());
    }
}
```

Exercício 25

```
package exer25;

public class Exer25 {
  public static void main(String[] args) {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Processamento donwload " + i + "%");
    }
    for (int i = 0; i <= 500; i++) {
      System.out.println("Atualizando a lista preço " + i + "%");
    }
    for (int i = 0; i <= 500; i++) {
      System.out.println("Enviando relatorio financeiro " + i + "%");
    }
  }
}
```

Exercício 26

```
package exer26;

public class Donwload extends Thread {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Processamento donwload " + i + "%");
    }
  }
}
```

```
package exer26;

public class Lista extends Thread {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Atualizando a lista preço " + i + "%");
    }
  }
}
```

```
package exer26;

public class Relatorio extends Thread {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Enviando relatorio financeiro " + i + "%");
    }
  }
}
```

```
package exer26;

public class Exer26 {
  public static void main(String[] args) {
    Donwload t1 = new Donwload();
    Lista t2 = new Lista();
    Relatorio t3 = new Relatorio();
    t1.start();
    t2.start();
    t3.start();
  }
}
```

Exercício 27

```
package exer27;

public class Donwload implements Runnable {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Processamento donwload " + i + "%");
    }
  }
}
```

```
package exer27;

public class Lista  implements Runnable {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Atualizando a lista preço " + i + "%");
    }
  }
}
```

```
package exer27;

public class Relatorio  implements Runnable {
  public void run() {
    for (int i = 0; i <= 500; i++) {
      System.out.println("Enviando relatorio financeiro " + i + "%");
    }
  }
}
```

```
package exer27;

public class Exer27 {
  public static void main(String[] args) {
    Thread t1 = new Thread(new Donwload());
    Thread t2 = new Thread(new Lista());
    Thread t3 = new Thread(new Relatorio());
    t1.start();
    t2.start();
    t3.start();
  }
}
```