

# Exercícios para o curso de Lógica e Programação Java

**Desenvolvidos por Fernando Anselmo  
Versão 2.1**

## Dia 01 – Conceitos Básicos

Dúvidas? Veja Slide 01 e Vídeo 01

### Exercício 1. Primeira Classe

- Abrir o BlueJ
- Criar um projeto
- Criar uma classe para mostrar seu nome (com base no exemplo realizado em sala).



Dica: Exemplo da classe realizada em sala:

```
public class Exemplo01 {  
    public static void main(String [] args) {  
        System.out.println("Olá mundo!!!");  
    }  
}
```

## Dia 02 – Tipos Primitivos e Entrada de Dados

Dúvidas? Veja Slide 02 e Vídeos 02, 03, 04 e 05

### Exercício 1. Sair do método main

- Modificar a classe para mostrar a soma de dois números quaisquer
- Criar um método chamado **executar()** com a seguinte assinatura:

```
public void executar() {  
}
```

- Colocar as instruções do método principal (**main**) neste método.
- Fazer o método principal (**main**) chamar este método.



Dica: Criar um objeto da própria classe.

### Exercício 2. Obter valores

Obter uma entrada pelo teclado contendo 2 números e como resultado mostrar:

```
Os números [primeiro número] e [segundo número]:  
Somados são [valor da soma]  
Subtraídos são [valor da subtração]  
Multiplicados são [valor da multiplicação]  
Divididos são [valor da divisão]
```



Dica: Para trabalhar com um objeto da classe *java.util.Scanner*:

- Para importar a classe: `import java.util.Scanner;`
- Para criar um objeto: `Scanner sc = new Scanner(System.in);`
- Para obter um tipo **int**: `int i = sc.nextInt();`
- Para obter um objeto **String**: `String s = sc.nextLine();`
- Para obter um tipo **float**: `float f = sc.nextFloat();`

## Dia 03 – Orientação a Objetos

Dúvidas? Veja Slide 03 e Vídeos 06, 07, 08 e 09

### Exercício 1. Trabalho de Arquiteto

- Acessar o site que contém os arquivos do curso.
- Na seção "Projetos para realizar", acessar o **Projeto 03 – Levantamento para o exercício de Orientação a Objetos**.
- Descrever o diagrama de classes conforme ensinado em sala.

## Dia 04 – Comandos

Dúvidas? Veja Slide 04

### Exercício 1. Comandos de Decisão

- Acessar o site que contém os arquivos do curso.
- Na seção "Projetos para realizar", acessar o **Projeto 02 – Música para o exercício de Comandos**.

### Exercício 2. Comandos de Repetição

- Criar uma classe chamada **Quadrilatero**.
- Criar um método chamado **executar()**. Solicitar um valor inteiro através de um objeto da classe *java.util.Scanner*. Com base neste valor desenhar um quadrilátero com asteriscos.
- Chamar o método **executar()** através do método principal (main) da seguinte forma:

```
public static void main(String [] args) {  
    Quadrilatero quad = new Quadrilatero();  
    quad.executar();  
}
```



Dica: Por exemplo, se o valor informado for 2, a saída será:

```
**  
**
```

Se o valor informado for 3, a saída será:

```
***  
***  
***
```

E assim sucessivamente.

## Dia 05 – Classe Gráfica em Java2D

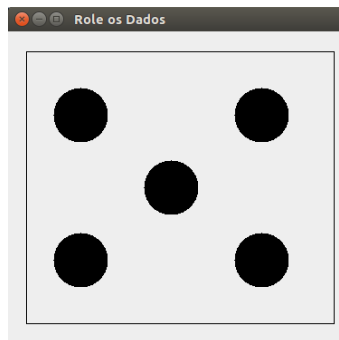
### Exercício 1. Desenho Lógico

Em uma janela, criar o seguinte desenho:



### Exercício 2. Controle de Primitivas

Simular o lance de um dado e mostrá-lo em uma janela gráfica.



## Dia 06 – Array e Classe java.lang.Math

### Exercício 1. Troca de Valores em um Array

Gerar 10 valores aleatórios (entre 1 e 10) em um array, e apresentá-los em ordem Crescente e Decrescente.

### Exercício 2. Percorrer Arrays Multidimensionais

- a) Dado o seguinte array multidimensional, que contém as entradas da agenda telefônica:

```
String [][] entradas =  
    {{ "Florence", "735-1234", "Manila"},  
      { "Joyce", "983-3333", "Quezon City"},  
      { "Becca", "456-3322", "Manila"} };
```

- b) Mostrar todos os dados contidos neste array da seguinte forma:

Nome - Telefone - Cidade

## Exercício 3. Trabalhando com Randômicos

a) Implementar o jogo CRAPS. Regra para UMA partida de CRAPS:

- Realizar o lançamento de dois dados:
  - ◆ Se o valor dos dados totalizar **7** ou **11** o jogador ganha
  - ◆ Se o valor dos dados totalizar **2**, **3** ou **12** o jogador perde
  - ◆ Se o valor dos dados totalizar outro valor (4, 5, 6, 8, 9, 10) este valor é preso. Deve repetir o lançamento dos dados até que:
    - Novamente o valor dos dados totalizar o valor que foi guardado, então o jogador ganha.
    - Se o valor dos dados totalizar **7**, então o jogador perde.

b) Fazer uma classe para jogar 10.000 partidas de CRAPS e mostrar quantas vezes o jogador ganhou e quantas perdeu.



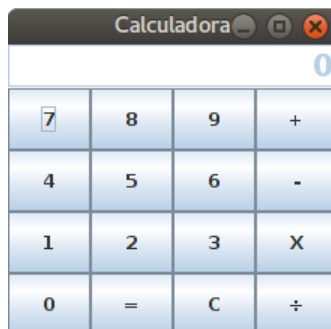
DICA: Para simular o lançamento de um dado usar o seguinte código:

```
byte lancamento = (byte) (Math.random()*6+1);
```

## Dia 07 – Pacote javax.swing e eventos

### Exercício 1. Calculadora

Criar a seguinte janela:



a) Trabalhe com 2 Arrays de elementos **JButton**, o primeiro para os botões numéricos e o segundo para os botões de função.

b) Organizar o objeto **visor** (JTextField), com o comando:

```
Font font1 = new Font("SansSerif", Font.BOLD, 20);  
visor.setFont(font1);  
visor.setHorizontalAlignment(JTextField.RIGHT);  
visor.setEnabled(false);  
visor.setEnabled(false);
```



Dica: Use a seguinte classe base

```
import javax.swing.*;  
import java.awt.*;  
  
class Calculadora extends JFrame {  
    public Calculadora() {  
        this.setTitle("Calculadora");  
    }  
}
```

## Exercícios para o Curso de Lógica e Programação Java

```
        this.setSize(220,220);
        // Inserir os elementos de criação da janela aqui
        this.setVisible(true);
    }
    public static void main(String [] args) {
        new Calculadora();
    }
}
```

### Exercício 2. Funcionamento da Calculadora

O funcionamento das calculadoras tradicionais é realizada da seguinte forma: digitar o primeiro valor (por exemplo **56**) em seguida a operação a ser realizada (por exemplo **+**), o segundo valor (por exemplo **46**), uma próxima operação ou o sinal de igualdade (no exemplo, ao ser realizada essa ação a soma dos valores deve aparecer no Visor).

a) No **primeiro exercício**, adicionar os seguintes atributos na classe:

```
private double total = 0.0; // Realizar os cálculos
private String numStr = ""; // Montar o número para o Visor
private double num = 0.0; // Guardar o primeiro valor digitado
private char ultAcao = 'I'; // Guardar a próxima ação a realizar
```

b) E os métodos para dar vida a calculadora:

#### Método para o Evento dos Números

```
private void montar(byte i) {
    numStr += i;
    visor.setText(numStr);
}
```

#### Método para as Teclas de Funções

```
private void calcular(char c) {
    if (c == 'C') {
        ultAcao = 'I';
        num = 0.0;
        total = 0.0;
        numStr = "";
    } else {
        if (numStr.length() > 0) {
            num = Double.parseDouble(numStr);
            numStr = "";
        }
        switch (ultAcao) {
            case 'I': total = num; break;
            case '+': total += num; break;
            case '-': total -= num; break;
            case 'X': total *= num; break;
            case '/': total /= num; break;
        }
        ultAcao = c;
    }
    visor.setText("" + total);
}
```

c) Chamar esses métodos através de eventos de ação colocados nos botões

Dica: Montagem do evento aoAcionar:

```
objeto.addActionListener(new ActionListener() {
```

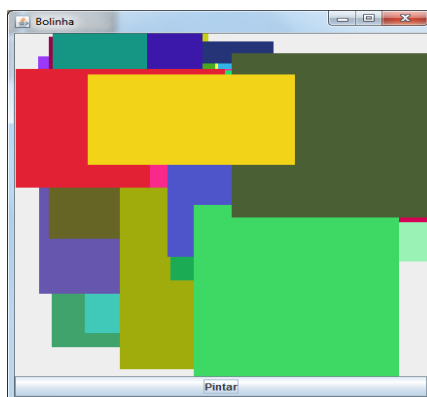


```
public void actionPerformed(ActionEvent e) {  
    // Método para realizar o procedimento  
}  
});
```

## Dia 08 – Eventos e classe java.util.Random

### Exercício 1. Combinar Eventos com Primitivas

Em uma janela gráfica, mostrar 50 quadriláteros em posições, tamanhos e cores aleatórias:



Dica: Montagem do evento aoAcionar:

```
objeto.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        // Método para realizar o procedimento  
    }  
});
```

### Exercício 2. Atualização da Janela de Primitivas

No segundo exercício do **Dia 05**, modificar a janela do dado para conter um botão que ao ser acionado, gerar randomicamente e mostrar um novo valor para o dado.

## Dia 09 – Aperfeiçoando o uso de Arrays

### Exercício 1.

Nesta apostila, realizar o projeto **Crescendo Letras**.

### Exercício 2.

Nesta apostila, realizar o projeto **Companhia Aérea**.

## Dia 10 – Proteção e Interface

### Exercício 1. Projeto e Implementação em OO

a) Descrever um diagrama de classes para um Sistema de Reserva e Ocupação de

Quartos de um hotel.

- b) Este projeto deve armazenar as reservas realizadas por um funcionário de um ou mais quartos para um determinado cliente.
- c) O funcionário deve ser capaz de:
  - o Verificar se um quarto está ocupado ou não
  - o Inserir ou alterar os dados de um cliente
  - o Realizar a reserva de um quarto para um cliente.
- d) Considerar os atributos para as classes como particulares.
- e) O cadastro de Cliente deve no mínimo possuir:
  - o Nome, Endereço e Telefone.
  - o O cliente ainda deve possuir um campo que armazene a quantidade de ocupações já realizadas por ele neste hotel.
- f) Um Quarto pode ser simples ou luxo e deve indicar o número de camas e o tipo de cada uma delas (solteiro ou casal).

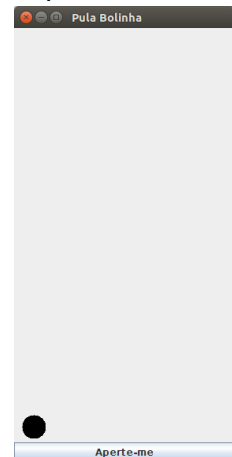


**Dica:** Deseja ser criativo? Então utilize interfaces e/ou classes abstratas para resolver esse problema.

## Dia 11 – Threads

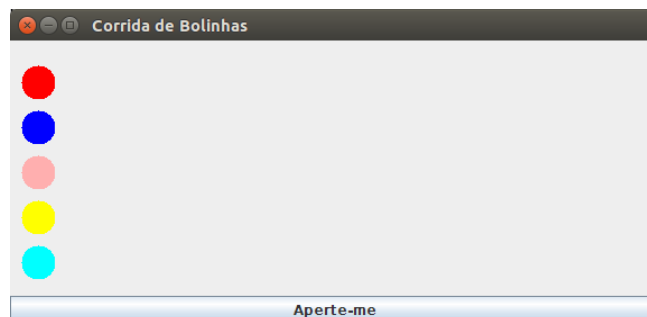
### Exercício 1. Movimento único

- a) Criar uma janela contendo uma bolinha no canto inferior esquerdo e um botão no rodapé da janela. Conforme a figura ao lado:
- b) Ao ser acionado o botão fazer a bolinha simular um salto até o topo da tela e terminar no canto inferior direito.



### Exercício 2. Movimento de vários objetos

- a) Criar cinco bolinhas coloridas, alinhadas à esquerda e um botão no rodapé na janela. Conforma a figura abaixo:



- b) Ao ser acionado o botão simular uma corrida entre as bolinhas até o final da janela.



## Dia 12 – java.util.Date, Arrays e Eventos

### Exercício 1. Manipulação de Data

- Obter um valor de 1 a 12 que representa um mês e um valor para o ano.
- Com base nesses valores montar um Calendário, por exemplo, entrada com mês **12** e ano **2012**, resultará em:

D	S	T	Q	Q	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					



Dica: Criar um array bidimensional para representar o calendário, por exemplo:

```
byte calendario = new byte[6][7];
```

- Descobrir quantos dias tem o mês
- Descobrir em qual dia da semana cai o dia 1º

### Exercício 2. Arrays e Eventos

Nesta apostila, realizar o projeto **Xadrez**.

## Dia 13 – Collections

### Exercício 1. Armazenar em Memória

- Criar uma tela para o armazenamento de dados dos Funcionários em memória conforme a seguinte figura:

The screenshot shows a window titled "Cadastro de Funcionário" with three input fields: "Matrícula:", "Nome:", and "Salário:". A "Gravar" button is positioned at the bottom right.

- Ao sair do campo **Matrícula**, caso não exista o funcionário armazenado ao ser acionado o botão **Gravar**, proceder a inclusão do registro.
- Ao sair do campo **Matrícula**, caso exista o funcionário guardado, mostrar os campos **Nome** e **Salário**, e ao ser acionado o botão **Gravar**, proceder a alteração do registro.

## Dia 14 – Pacote java.io (classe File e tipo char)

### Exercício 1. Consultar Arquivos

Mostrar todos os arquivos encontrados em uma determinada pasta através da classe *java.io.File*.



Dica: Lembramos que para obter uma lista de arquivos de uma pasta:

```
File file = new File(nomePasta);
String[] fileNames = file.list();
File[] files = file.listFiles();
```

### Exercício 2. Bloco de Notas

a) Criar uma classe chamada **MeuBloco** que representa uma janela para implementar um Bloco de Notas

b) Utilizar o seguinte código no construtor para criar essa janela:

```
private JTextArea area = new JTextArea();

public MeuBloco() {
    super("Meu Editor");
    this.setSize(300, 300);
    JButton btGravar = new JButton("Gravar");
    btGravar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Método para gravar
        }
    });
    JButton btCarregar = new JButton("Carregar");
    btCarregar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Método para carregar
        }
    });
    JPanel pn = new JPanel();
    pn.add(btGravar);
    pn.add(btCarregar);
    this.add(pn, BorderLayout.NORTH);
    this.add(new JScrollPane(area));
    this.setVisible(true);
}
```

c) Usar os seguintes códigos para gravar os dados no arquivo:

```
FileDialog dig = new FileDialog(this, "Salvar Arquivo", FileDialog.SAVE);
dig.setVisible(true);
String arq = dig.getDirectory() + dig.getFile();
FileWriter fw = new FileWriter(arq);
fw.write(area.getText());
fw.close();
```

d) Usar os seguintes códigos para ler os dados do arquivo:

```
FileDialog dig = new FileDialog(this, "Abrir Arquivo", FileDialog.LOAD);
dig.setVisible(true);
String arq = dig.getDirectory() + dig.getFile();
BufferedReader bf = new BufferedReader(new FileReader(arq));
```

```
String linha = "";  
while((linha = bf.readLine()) != null)  
    area.append(linha + '\n');  
bf.close();
```

## Dia 15 – Pacote java.io (tipo byte)

### Exercício 1. Uso da Rede

Implementar uma rede **CHAT** em Java. Ver as classes Client/Server no final dessa apostila (quais possibilidades de projetos você consegue pensar para utilizar essas classes? Que tal um jogo de dominó na rede?)

### Exercício 2. Armazenar Objetos em Disco

No primeiro exercício do **Dia 13**, modificar a forma de armazenar os dados para gravá-los em disco no lugar da memória.

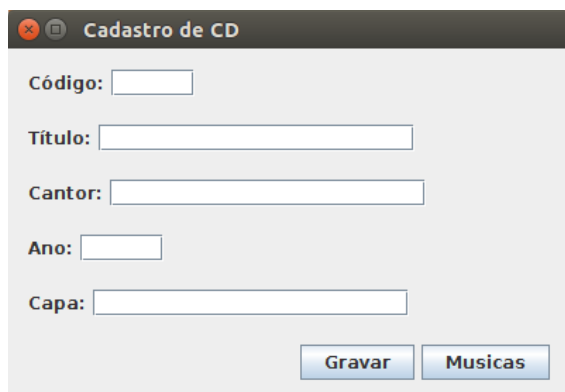
## Dia 16 – Pacote java.sql

### Exercício 1. Armazenar em Arquivo de Dados

Com base nos seguintes comandos de criação de tabelas:

```
CREATE TABLE cd (  
    codigo INTEGER NOT NULL,  
    titulo VARCHAR(50),  
    cantor VARCHAR(50),  
    ano INTEGER,  
    capa VARCHAR(50),  
    PRIMARY KEY (codigo))  
CREATE TABLE trilha (  
    codcd INTEGER NOT NULL,  
    numero INTEGER NOT NULL,  
    nome VARCHAR(50),  
    tempo VARCHAR(5),  
    PRIMARY KEY (codcd, numero))
```

Criar telas para proceder o cadastro de CD e Trilha, conforme a seguinte figura:



Cadastro de CD

Código:

Título:

Cantor:

Ano:

Capa:

Gravar Musicas



Cadastro de Trilha

Número:

Nome:

Tempo:

Gravar

Detalhes:

- Seu funcionamento é semelhante a janela de cadastro de Funcionário realizado no **Dia 13**.
- A janela de CD chamará a de cadastro de Trilhas, através do botão **Músicas**.
- Obter o conector JDBC para o banco de dados e disponibilizá-lo para o BlueJ conforme instruções dadas em sala.

## Dia 17 – Pacote java.sql

### Exercício 1. Consultar em um Arquivo de Dados

No primeiro exercício do **Dia 16**, adicionar uma tela para mostrar o CD cadastrado, conforme a seguinte figura:



Detalhes:

- Trabalhe com as capas de forma semelhante ao tratamento de imagens visto no Projeto **Xadrez**.
- Ao ser informado um código do CD e pressionar a tecla TAB para sair deste campo, o CD deve ser mostrado.

## Atividades Extras para o Conhecimento

Estes exercícios não serão realizados em sala, porém são excelentes para aperfeiçoar seu conhecimento, então tente executá-los e peça dicas ao professor (**Lembre-se que só se aprende, praticando**).

### Exercício 1. Math e Scanner

- Criar uma classe chamada **Triangulo**.
- Criar um método chamado **entrar()**. Solicitar os valores para **cateto1** e **cateto2** através de um objeto da classe `java.util.Scanner`.
- Calcular o valor da hipotenusa através da seguinte fórmula de Pitágoras:

$$\text{hipotenusa}^2 = \text{cateto}^2 + \text{cateto}^2;$$

d) Mostrar esses dados da seguinte forma:

Dados do seu Triângulo:  
Cateto 1: [valor do cateto 1]  
Cateto 2: [valor do cateto 2]  
Hipotenusa: [valor da hipotenusa]

e) Chamar o método `entrar()` através do principal (**main**) da seguinte forma:

```
public static void main(String [] args) {  
    Triangulo triangulo = new Triangulo();  
    triangulo.entrar();  
}
```

## Exercício 2. Percorrer um Array

a) Com base no seguinte array de Strings, inicializado com os nomes dos dias da semana.

```
String dias[] = {"Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado",  
"Domingo"};
```

b) Mostrar todo o conteúdo deste array. Use inicialmente um laço **while**. Faça o mesmo para os laços **do**, **for** e **foreach**.

## Exercício 3. Localização em Array

a) Armazenar em um array até o 20 termo da sequência de Fibonacci.

b) Obter do usuário o valor de 1 a 20 e com base neste valor, localizar e mostrar o termo correspondente na sequência de Fibonacci<sup>1</sup>.

## Exercício 4. Ordenação em Array

a) Criar um array com 10 palavras quaisquer. Por exemplo:

```
String [] palavras = new String[]{"Pal1", "Pal2", ..., "Pal10"};
```

b) Ordenar lexicograficamente<sup>2</sup> este array.

**Dicas:** Para trocar duas palavras de posição em um Array, por exemplo, posições p1 e p2:

```
String aux = palavras[p1];  
palavras[p1] = palavras[p2];  
palavras[p2] = aux;
```

## Exercício 5. Projeto OO

a) Em um Diagrama de Classes

b) Criar uma classe chamada **Livro**.

<sup>1</sup> Leonardo Fibonacci (Pisa, 1170 a 1250) foi um matemático italiano, tido como o primeiro grande matemático europeu do Medievo. A Sequência de Fibonacci consiste em uma sucessão de números, tais que, definindo os dois primeiros números da sequência como 0 e 1, os números seguintes serão obtidos por meio da soma dos seus dois antecessores.

<sup>2</sup> Ordem Lexicográfica é a ordem que as palavras aparecem no Dicionário. Por exemplo: Abacate, Abacateiro, Abacaxi, Abacial, Ábaco, ...



- c) Criar uma classe chamada **LivroDeLivraria** que contém os dados básicos de um livro que está à venda em uma livraria.
- d) Criar uma classe chamada **LivroDeBiblioteca** que contém os dados básicos de um livro em uma biblioteca, que pode ser emprestado a leitores.
- e) Criar outras classes que sejam necessárias para implementar os modelos de **Livraria** e **Biblioteca**. Com funções respectivas para vender e emprestar livros.
- f) Implementar todo o Diagrama de Classe em linguagem Java.

## Exercício 6. Projeto OO

- d) Implementar uma classe em linguagem Java, conforme o seguinte Diagrama de Classes abaixo:

```
+-----+
| Triangulo |
+-----+
| - cateto1: float |
| - cateto2: float |
+-----+
| + setCateto1(cateto1: float) |
| + setCateto2(cateto2: float) |
| + getCateto1(): float |
| + getCateto2(): float |
| + obterHipotenusa(): float |
+-----+
```

- e) Os métodos set/get possuem a implementação padrão segundo o **Princípio do Encapsulamento**.
- f) O método **obterHipotenusa()** retorna o valor da hipotenusa com base na seguinte fórmula de Pitágoras:

$$\text{hipotenusa}^2 = \text{cateto}^2 + \text{cateto}^2;$$

- g) Implementar a seguinte classe, conforme o modelo UML de Classe, abaixo:

```
+-----+
| Matematico |
+-----+
| |
+-----+
| + main(args: String []) |
| + entrar() |
+-----+
```

- h) O método **entrar()** contém:
  1. A solicitação dos valores do cateto1 e cateto2.
  2. Com esses dados criar um objeto da classe **Triangulo**.
  3. E mostrá-lo na tela da seguinte forma:

Dados do Triângulo:  
Cateto 1: [valor do cateto 1]  
Cateto 2: [valor do cateto 2]

Hipotenusa: [valor da hipotenusa]

- i) O método **main()** contém a chamada para o método **entrar()**.

## Exercício 7. java.io (Classe File)

Mostrar em qual pasta se encontra um determinado arquivo no Sistema Operacional (não considerar que este arquivo pode possuir cópias).

## Exercício 8. Conversão de Valores

- a) Ler uma distância expressa em milhas e convertê-la para quilômetros.
- b) Ler uma quantia expressa em dólares e convertê-la para reais.
- c) Ler um valor em segundos e converter para o formato hh:mm:ss.
- d) Sabendo que 1 Kg na Terra corresponde a: 0,37 Kg em Mercúrio; 0,88 Kg em Vênus; 0,38 Kg em Marte; 2,64 Kg em Júpiter; 1,15 Kg em Saturno; 1,17 Kg em Urano; 1,18 Kg em Netuno e 0,11 Kg em Plutão<sup>3</sup>. Receber o peso do usuário e mostrar seu correspondente em cada um dos planetas.
- e) Ler uma distância expressa em milhas terrestres e convertê-la para milhas náuticas.



Dica: Para as fórmulas de conversão, considere:

1 milha = 1,609 km

1 dólar = 2,05 reais

12 milhas terrestres = 19,308 km e 12 milhas náuticas = 22,224 km

## Exercício 9. Aplicação do Princípio do Encapsulamento

Ler o valor do salário de um funcionário e o percentual de aumento. Calcular e mostrar o valor do aumento e o novo salário.

## Exercício 10. Verificar atributos

Ler um número através do teclado e mostrar se é um número Par ou Ímpar. Um número é considerado Par se for divisível por 2.

## Exercício 11. Conversão de String para Tipo Primitivo

Ler um número e mostrar a tabuada completa para esse número:

num X 1 = resultado

num X 2 = resultado

...

num X 10 = resultado

## Exercício 12. Laços de Repetição

- a) Ler um número e mostrar se é ou não um número primo.
- b) Mostrar os 50 primeiros números primos<sup>4</sup>.

<sup>3</sup> Plutão não é mais considerado um planeta. Mas, como o assunto ainda gera controvérsias, optei por incluí-lo.

<sup>4</sup> Números primos são números divisíveis somente por 1 ou por ele mesmo.

## Exercício 13. Collections

a) Implementar a classe conforme o seguinte Diagrama de Classe abaixo:

```
+-----+
| Questao                               |
+-----+
| - numero: int                         |
| - pergunta: String                   |
| - opcao1: String                     |
| - opcao2: String                     |
| - opcao3: String                     |
| - opcao4: String                     |
| - resposta: byte                     |
+-----+
| + setNumero(numero: int)             |
| + setPergunta(pergunta: String)     |
| + setOpcao1(opcao1: String)          |
| + setOpcao2(opcao2: String)          |
| + setOpcao3(opcao3: String)          |
| + setOpcao4(opcao4: String)          |
| + setResposta(resposta: byte)        |
| + getNumero(): int                   |
| + getPergunta(): String              |
| + getOpcao1(): String                |
| + getOpcao2(): String                |
| + getOpcao3(): String                |
| + getOpcao4(): String                |
| + getResposta(): byte                |
+-----+
```

b) Criar uma classe chamada **Simulado**.

c) Nesta classe criar um atributo da classe *java.util.ArrayList* para conter um conjunto de objetos de **Questao**.

d) Criar um método chamado **cadastrar()**, solicitar o número, a questao, o valor da opção 1, o valor da opção 2, o valor da opção 3, o valor da opção 4 e qual a resposta correta. Com esses dados criar um objeto da classe **Questao** e armazená-lo no objeto criado no item anterior.

e) Criar um método chamado **examinar()**, selecionar e mostrar uma questão aleatoriamente entre todas questões que foram armazenadas previamente, da seguinte forma:

Questão [campo numero]

[campo pergunta]

1. [campo opcao1]
2. [campo opcao2]
3. [campo opcao3]
4. [campo opcao4]

Sua resposta é:

f) Ainda no método **examinar()**, obter a resposta do usuário e verificar se é ou não correta. Compare-a com o campo resposta da questão selecionada.

g) Criar um método chamado **menu()**, com a seguinte codificação:



## Exercícios para o Curso de Lógica e Programação Java

```
char opc = 'c';
do {
    System.out.println("Suas Opções:");
    System.out.println("  (a) Cadastrar Questões");
    System.out.println("  (b) Realizar Prova");
    System.out.println("  (c) Sair");
    opc = sc.next().charAt(0);
    switch (opc) {
        case 'a': cadastrar(); break;
        case 'b': examinar(); break;
    }
} while (opc != 'c');
```

h) Criar o método principal (main) para chamar o método **menu()**.



**Dica:** Para usar um **ArrayList**:

Importar a classe: `import java.util.ArrayList;`

Criar a coleção: `ArrayList<[NomeClasse]> lst = new ArrayList<[NomeClasse]>();`

Adicionar um elemento: `lst.add(objeto);`

Obter quantos elementos a coleção possui: `int qtd = lst.size();`

Obter um determinado elemento: `[NomeClasse] qst = lst.get(pos);`

Mostrar todos elementos cadastrados:

```
for ([NomeClasse] obj: lst)
    System.out.println(obj.getCampo1() + " " + obj.getCampo2() + ...);
```

## Exercício 14. Controle de Coleções

a) Criar uma classe chamada **Produto**, para implementar os dados de um produto com os seguintes atributos:

```
private String descricao;
private float preco;
```

b) Para a classe **Produto**, criar métodos modificadores com assinaturas padrão SET/GET.

c) Criar uma classe chamada **Fornecedor**, para armazenar os produtos em um objeto da classe `java.util.ArrayList`.

d) Criar um método chamado **entrar()**. Solicitar a descrição e o preço de um Produto através de um objeto da classe `java.util.Scanner`.

e) Com esses dados criar um objeto da classe **Produto** e armazená-lo.

f) Criar um método chamado **listarProdutos()**. Mostrar todos os produtos que foram adicionados armazenados.

g) Criar um método chamado **menu()**, com a seguinte codificação:

```
char opc = 'c';
do {
    System.out.println("Suas Opções:");
    System.out.println("  (a) Cadastro de Produtos");
    System.out.println("  (b) Listagem de Produtos");
    System.out.println("  (c) Sair");
    opc = sc.next().charAt(0);
}
```

## Exercícios para o Curso de Lógica e Programação Java

```
switch (opc) {
    case 'a': entrar(); break;
    case 'b': listarProdutos(); break;
}
} while (opc != 'c');
```

- h) Para a classe Fornecedor, criar o método principal (main) para chamar o método menu().

### Exercício 15. Uso de StringBuilder ou StringBuffer

Obter uma palavra e retornar se é ou não um Palíndromo<sup>5</sup>

### Exercício 16. Uso dos métodos das String

- a) Obter uma frase do usuário.  
b) Percorrer esta frase e retornar qual a quantidade de vogais, espaços em brancos e consoantes.



Dica: Use o método **charAt(posição)**

### Exercício 17. Troca de Letras

As mensagens secretas foram inventadas durante as grandes guerras mundiais para, caso o mensageiro fosse pego a mensagem não poderia ser lida.

- a) Obter uma palavra, cujas letras devem estar no intervalo de A a Z. Trocar essas letras através do seguinte modelo:

A → Q	B → W	C → E	D → R	E → T	F → Y	G → U
H → I	I → O	J → P	K → A	L → S	M → D	N → F
O → G	P → H	Q → J	R → K	S → L	T → Z	U → X
V → C	X → V	Y → B	W → N	Z → M		

Ou seja: **FERNANDO** resultará em **YTKFQFRG**.

- b) Obter uma palavra com as letras trocadas com base no modelo anterior e transformá-la nas letras corretas.

Ou seja: **YTKFQFRG** resultará em **FERNANDO**.

### Exercício 18. Troca de Letras (opção 2)

Obter uma frase e trocar as letras por sua sequência correspondente no Código Morse<sup>6</sup>. As letras são representadas da seguinte forma:

A • -	B - • • •	C - • - •	D - • •	E •
F • • - •	G - - •	H • • • •	I • •	J • - - -

<sup>5</sup> Palíndromo é uma palavra ou frase que pode ser lida da esquerda pra direita ou da direita pra esquerda obtendo-se a mesma palavra. Ex.: ANA, ARARA, OSSO.

<sup>6</sup> Em maio de 1844, Samuel F. B. Morse enviou a seguinte mensagem “What hath God wrought!” através de um telegrafo de Washington a Baltimore. O código Morse foi desenhado para codificar letras em símbolos, uma sequência de tons curtos e longos, tradicionalmente chamados de pontos e traços.

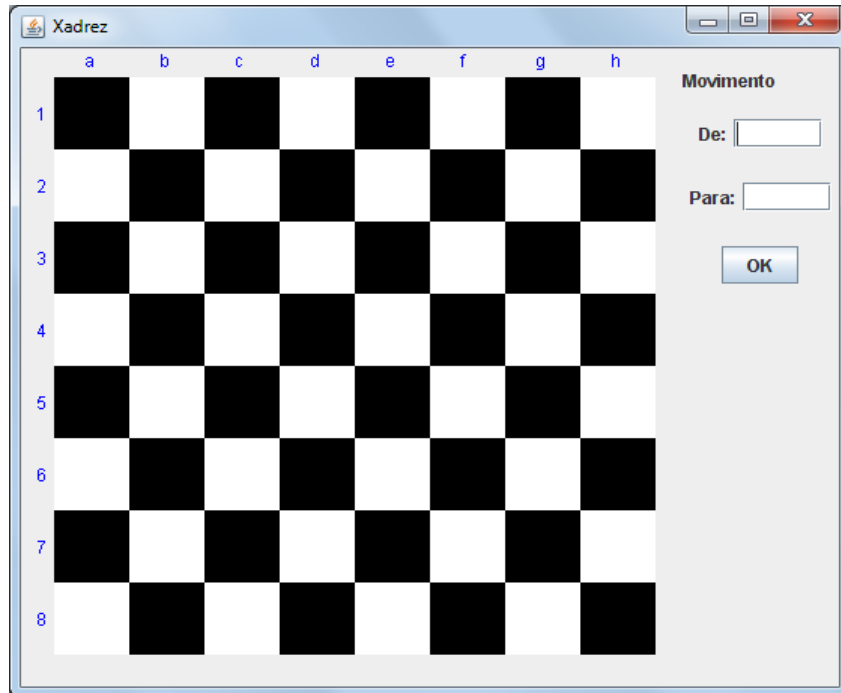
Exercícios para o Curso de Lógica e Programação Java

K - • -      L • - • •      M - -      N - •      O - - -  
P • - - •      Q - - • -      R • - •      S • • •      T -  
U • • -      V • • • -      W • - -      X - • • -      Y - • - -  
Z - - • •

# Projeto Jogo de Xadrez

## a. Construção da Janela

Construa a seguinte janela, utilizando os **layouts** e o método **paint()**:



## b. Colocando as peças nos seus lugares

Obtendo as peças com seu professor, coloque-as no tabuleiro na seguinte ordem:



em "a1" e "h1"



em "b1" e "g1"



em "c1" e "f1"



em "d1"



em "e1"



em "a2" a "h2"



em "a8" e "h8"



em "b8" e "g8"



em "c8" e "f8"



em "d8"



em "e8"



em "a7" e "h7"



**Dica:** Para inserir a imagem use a seguinte instrução no método paint:

```
g.drawImage(Toolkit.getDefaultToolkit().getImage("pecas/AB.gif"), 50,50, null);
```

## c. Realizando o movimento

Realize o movimento através dos dois JTextField de entrada "de" e "para", informando a posição que se encontra a peça e para aonde ela deve ir. Ex: de: e2 para: e4.

Simule o movimento na tela.

## Projeto Crescendo Letras

### Definição

Criar um projeto que receba uma cadeia de caracteres (String) através do argumento passado pelo método principal (main) e como saída substituir cada uma das letras da cadeia conforme o seguinte alfabeto:

```
*****  ****  *****  ****  *****  *****  ****  *  *  *****  *****  *  *  *  *  *  *  *****  *****
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*****  *****  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
```

Não utilizar quaisquer caracteres especiais, números, letras minúsculas ou acentuação, utilizar somente letras maiúsculas de A até Z. Como por exemplo:

```
java Letras DEVER
```

Que resultará na seguinte saída:

```
****  *****  *  *  *****  *****
*  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *
****  *****  *  *****  *  *  *
```

## Projeto Companhia Aérea

### Da Venda de Passagens

A companhia AirLines Java deseja informatizar sua venda de passagens aéreas, por esse motivo, expôs as seguintes regras de negócio.

Cada avião da companhia contém 12 lugares, que são distribuídos da seguinte forma: 6 lugares são na janela e possuem número par e 6 lugares são no corredor e possuem número ímpar. Então o cliente informa qual assento deseja e seu nome, através da seguinte janela:

```
-----
Companhia AirLines Java - Venda de Passagens
-----
Voo Número 001 - Informe o assento de sua preferência:
    Na Janela (Disponíveis: 2 4 6 8 10 12)
    No Corredor (Disponíveis: 1 3 5 7 9 11)
Opção: [      ]
Cliente: [                                     ]
-----
```

Uma vez informado um assento disponível, mostrar a mensagem: "Passagem Emitida no Assento XX" e retirar o assento da listagem na tela. Caso seja informado um assento não disponível, mostrar a mensagem: "Assento ocupado! Favor observar

disponibilidade" e retornar para a tela.

### Da Venda Concluída

Uma vez que TODOS os assentos foram vendidos, informar a mensagem: "Voo confirmado, partida em 10 minutos" e mostrar a seguinte listagem conforme registrado no sistema:

-----  
Companhia AirLines Java - Voo Confirmado  
-----

Relação dos Passageiros para o Voo Número 001:

- 01. Nome do cliente informado
  - 02. Nome do cliente informado
  - 03. Nome do cliente informado
  - ...
  - 12. Nome do cliente informado
- 

Retornar a primeira tela, modificar o número do Voo e recomeçar a venda das novas passagens.

## Projeto Tali

Este projeto não será realizado em sala, porém é excelente para aperfeiçoar seu conhecimento, então tente executá-los e peça dicas ao professor (**Lembre-se que só se aprende, praticando**).

### a. Tabuleiro de Opções

Tali é um jogo criado na antiga Grécia, mas que foi difundido pelos Romanos, a versão moderna do jogo envolve a utilização de 5 dados e uma série de opções que devem ser escolhidas com base em lógica, as opções são as seguintes:

- a) Total de valores 1 (pontos corresponde ao somatório de 1s)
- b) Total de valores 2 (pontos corresponde ao somatório de 2s)
- c) Total de valores 3 (pontos corresponde ao somatório de 3s)
- d) Total de valores 4 (pontos corresponde ao somatório de 4s)
- e) Total de valores 5 (pontos corresponde ao somatório de 5s)
- f) Total de valores 6 (pontos corresponde ao somatório de 6s)
- g) 3 números com um mesmo valor (pontos corresponde ao somatório dos números iguais)
- h) 4 números com um mesmo valor (pontos corresponde ao somatório dos números iguais)
- i) 3 e 2 números com um mesmo valor (27 pontos)
- j) Sequencia pequena com valores de 1 a 5 (30 pontos)
- k) Sequencia grande com valores de 2 a 6 (40 pontos)
- l) 5 números com um mesmo valor (50 pontos)
- m) Somatório de todos os números (pontos corresponde ao somatório dos

números)

Nas opções de 1 a 6, se o somatório for maior que 62 pontos, recebe 35 pontos de bônus.

## b. Como jogar

As regras de Tali são:

1. Rola-se os 5 dados
2. Com o resultado, o usuário pode escolher qual opção deseja realizar, desde que haja opção aberta e com a combinação possível.
3. Se não houver nenhuma combinação possível para o resultado dos dados, o usuário deve escolher uma opção aberta para zerar.
4. Repetir esses passos 13 vezes (ou seja, ao final todas as opções terão seus valores ou zeradas)

## c. Passo 1: Para testar o jogo (Desenvolver a lógica)

Realizar a sequência de operações a seguir para desenvolver e testar a lógica do jogo, utilize conjunto de métodos `System.out.println()`:

Valor dos dados nessa jogada: ? ? ? ? ?

Suas opções são:

- a. Total de 1s:
- b. Total de 2s:
- c. Total de 3s:
- d. Total de 4s:
- e. Total de 5s:
- f. Total de 6s:
- g. 3 iguais:
- h. 4 iguais:
- i. 3 e 2 números iguais:
- j. Sequência pequena:
- k. Sequência grande:
- l. 5 iguais
- m. Somatório:

Qual a sua opção: [] ← Solicitar a informação do usuário

Repetir essa sequência 13 vezes, mostrando os resultados adquiridos conforme o usuário vai escolhendo suas opções e os dados são lançados. Observar as regras do Tali.

## d. Passo 2: Criar a aparência do Jogo (Classe Gráfica)

Criar uma janela para servir como tabuleiro do jogo, conforme a seguinte figura:

## Exercícios para o Curso de Lógica e Programação Java

Opções:	Valor dos Dados
a) 1s: 0	XX
b) 2s: 0	XX
c) 3s: 0	XX
d) 4s: 0	XX
e) 5s: 0	XX
f) 6s: 0	
g) 3 iguais: 0	
h) 4 iguais: 0	
i) 3 e 2 iguais: 0	
j) Seq.Pequena: 0	
k) Seq.Grande: 0	
l) 5 iguais: 0	
m) Somatório: 0	

Sua Opção:

OK



### Dicas:

1. Utilizar 350 pixels para a largura e 530 pixels para a altura da Janela.
2. Utilizar o método: `this.setLocationRelativeTo(null);` para a janela aparecer centralizada na tela.
3. Usar as seguintes instruções para criar uma borda em torno a janela:

```
JPanel bordaJanela = new JPanel();
bordaJanela.setLayout(new BorderLayout());
bordaJanela.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
this.setContentPane(bordaJanela);
```
4. Usar as seguintes instruções para criar a borda negra em torno ao valor dos dados:

```
JPanel pnBorda1 = new JPanel();
pnBorda1.setBorder(BorderFactory.createLineBorder(Color.black));
pnBorda1.add(dado1);
this.add(pnBorda1);
```
5. Criar somente os objetos necessários para o uso no jogo e utilizar objetos anônimos aos outros que compõem a janela.

### e. Passo 3: Associar a Lógica a Janela Gráfica

Tornar a janela descrita no passo anterior totalmente funcional.



# Mundos de OpenKarel

Dúvidas? Veja todo o projeto OpenKarel no site, na seção de projetos.

## Exercício 1.

- Acessar o site que contém os arquivos do curso.
- No projeto "OpenKarel", baixar a biblioteca e instalar no BlueJ conforme as instruções obtidas em sala.
- Realizar download do **Mapa 01 – Conhecer o Mundo de Karel**.
- Seu objetivo neste mapa é fazer Karel pegar o sinalizador e soltá-lo na última Rua.



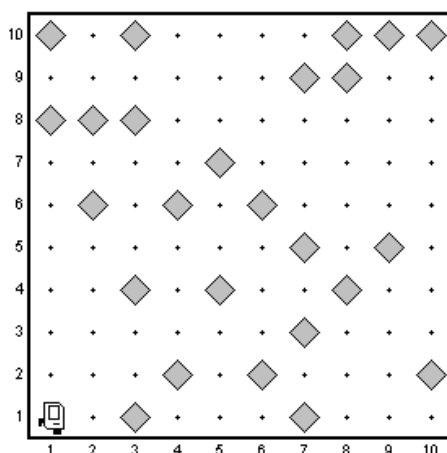
Dica: Criar um programa livre, imagine que o sinalizador pode estar um pouco mais a frente ou mais atrás, o alambrado pode ter dois ou três andares em vez de um só.

## Exercício 2.

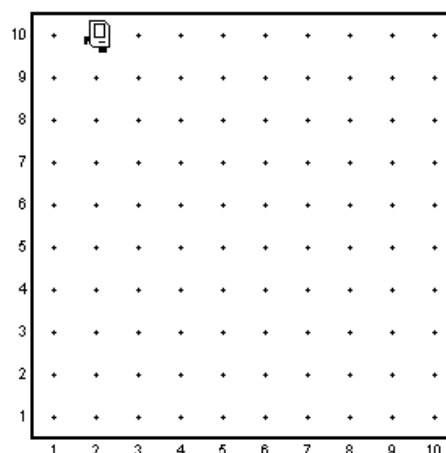
- Acessar o site que contém os arquivos do curso.
- No projeto "OpenKarel", realizar download do **Mapa 03 – Subir e Descer**.
- Seu objetivo neste mapa é fazer Karel subir e descer todos os obstáculos até chegar à última rua.

## Exercício 3.

- Acessar o site que contém os arquivos do curso.
- No projeto "OpenKarel", realizar download do **Mapa 05 – Limpar Sinais**.
- Seu objetivo neste mapa é fazer Karel pegar todos os sinalizadores que se encontram espalhados.



Antes



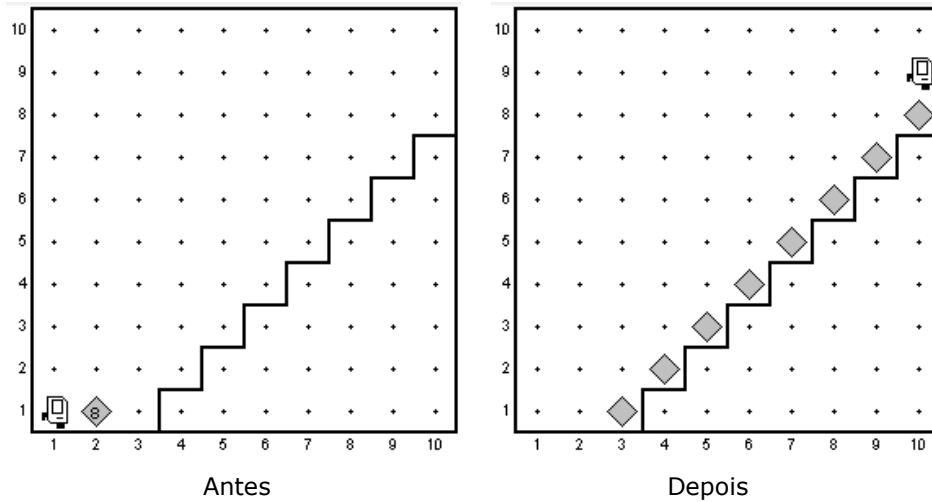
Depois

## Exercício 4.

- Acessar o site que contém os arquivos do curso.

## Exercícios para o Curso de Lógica e Programação Java

- b) No projeto "OpenKarel", realizar download do **Mapa 06 – Sinais na Escada**.
- c) Seu objetivo neste mapa é fazer Karel pegar todos os sinalizadores e colocar um em cada degrau da escada.



Dica: Karel está disposta a lhe ensinar muito mais sobre Lógica e Java, veja todos os vídeos dos **Desafios de Karel** que estão disponíveis no projeto.

# Chat em Java

## Classe para Implementar o Cliente

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.net.Socket;

public class ChatClient extends JFrame {

    private BufferedReader in;
    private PrintWriter out;
    private JTextField textField = new JTextField(40);
    private JTextArea messageArea = new JTextArea(8, 40);

    public ChatClient() {
        super("CHAT Cliente");
        this.setSize(300,300);
        textField.setEditable(false);
        messageArea.setEditable(false);
        this.add(textField, BorderLayout.NORTH);
        this.add(new JScrollPane(messageArea), BorderLayout.CENTER);
        textField.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                enviarMensagem();
            }
        });
        this.setVisible(true);
    }
}
```

## Exercícios para o Curso de Lógica e Programação Java

```
private void enviarMensagem() {
    out.println(textField.getText());
    textField.setText("");
}
private String getServerAddress() {
    return JOptionPane.showInputDialog(
        this, "Digite o Endereço IP do Servidor:",
        "Chat em Java", JOptionPane.QUESTION_MESSAGE);
}
private String getNameChat() {
    return JOptionPane.showInputDialog(
        this, "Digite seu Nome:",
        "Chat em Java", JOptionPane.QUESTION_MESSAGE);
}
private void run() {
    try {
        String serverAddress = getServerAddress();
        Socket socket = new Socket(serverAddress, 9001);
        in = new BufferedReader(new InputStreamReader(
            socket.getInputStream()));
        out = new PrintWriter(socket.getOutputStream(), true);
        String line;
        do {
            line = in.readLine();
            if (line.startsWith("SUBMITNAME")) {
                out.println(getNameChat());
            } else if (line.startsWith("NAMEACCEPTED")) {
                textField.setEditable(true);
            } else if (line.startsWith("MENSAGEM")) {
                messageArea.append(line.substring(8) + "\n");
            }
        } while (!line.substring(line.length()-
            5, line.length()).equals("ADEUS"));
    } catch (IOException e) {
        e.printStackTrace();
    }
    JOptionPane.showMessageDialog(this, "Até a próxima",
        "Chat em Java", JOptionPane.INFORMATION_MESSAGE);
    System.exit(0);
}
public static void main(String[] args) throws Exception {
    new ChatClient().run();
}
}
```

## Classe para Implementar o Servidor

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashSet;

public class ChatServer {

    private static final int PORT = 9001;
    private static HashSet<String> names = new HashSet<String>();
    private static HashSet<PrintWriter> writers = new HashSet<PrintWriter>();

    public static void main(String[] args) throws Exception {
```

## Exercícios para o Curso de Lógica e Programação Java

```
System.out.println("O Servidor se encontra rodando...");
ServerSocket listener = new ServerSocket(PORT);
try {
    while (true) {
        new Handler(listener.accept()).start();
    }
} finally {
    listener.close();
}
}

private static class Handler extends Thread {
    private String name;
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;

    public Handler(Socket socket) {
        this.socket = socket;
    }

    public void run() {
        try {
            in = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);
            while (true) {
                out.println("SUBMITNAME");
                name = in.readLine();
                if (name == null) return;
                synchronized (names) {
                    if (!names.contains(name)) {
                        names.add(name);
                        break;
                    }
                }
            }
            out.println("NAMEACCEPTED");
            writers.add(out);
            while (true) {
                String input = in.readLine();
                if (input == null) return;
                for (PrintWriter writer : writers) {
                    writer.println("MENSAGEM " + name + ": " + input);
                }
            }
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            if (name != null) { names.remove(name); }
            if (out != null) { writers.remove(out); }
            try { socket.close();
            } catch (IOException e) {
            }
        }
    }
}
}
```