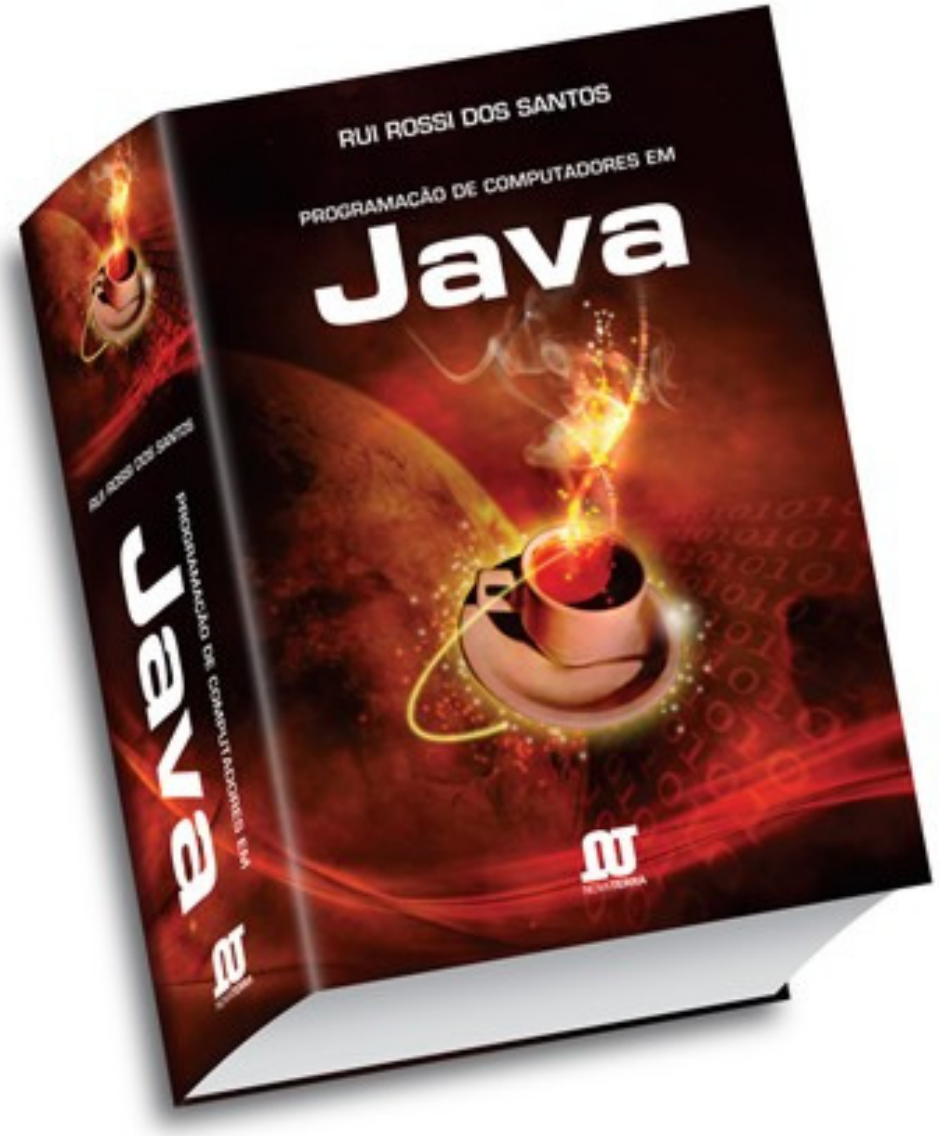


Capítulo 32

Programação Multi-Thread

Objetivos do Capítulo

- Analisar as operações que podem ser realizadas com a linha de execução principal de um programa.
- Indicar como criar múltiplas linhas de execução e como executá-las paralelamente.
- Apresentar o mecanismo que pode ser empregado para sincronizar um método e evitar que múltiplas linhas de execução o acessem ao mesmo tempo.

Introdução

- **Linha de execução**
  - **Ponteiro que orienta a execução de um programa**
  - **Objeto da classe java.lang.Thread**

- **Considerações importantes:**
  - **Todo programa possui uma linha de execução: main**
  - **É possível criar linhas de execução secundárias**
  - **A execução de múltiplas linhas de execução usa escalonamento**

- **Informações de uma linha de execução**
  - **Nome**
  - **Prioridade**

Introdução

- java.lang.Thread implements java.lang.Runnable
  - Atributos estáticos
    - int MAX_PRIORITY
    - int MIN_PRIORITY
    - int NORM_PRIORITY
  - Construtures
    - Thread( )
    - Thread(Runnable target)
    - Thread(String name)
    - Thread(Runnable target, String name)

Introdução

- java.lang.Thread implements java.lang.Runnable
  - Métodos estáticos
    - Thread currentThread( )
    - sleep(long millis)
  - Outros métodos
    - String getName( )
    - int getPriority( )
    - setName(String name)
    - setPrioriry(int newPriority)
    - run( )
    - start( )

- java.lang.Runnable
  - Método
    - run( )

Introdução

- **Criação de uma linha de execução**
  - **Primeira forma:**
    - **Criar uma nova classe derivada da classe Thread**
    - **Sobrescrever seu método run( )**
    - **Criar um novo objeto a partir dela**
  - **Segunda forma:**
    - **Criar uma nova classe que realize a interface Runnable**
    - **Implementar o método run( )**
    - **Criar um novo objeto dessa classe**
    - **Utilizá-lo na invocação do construtor da classe Thread**

Introdução

- **Linha corrente e pausa**

```
Thread th = Thread.currentThread( );
try {Thread.sleep(1000);}
catch (InterruptedException iex) {}
```

- **Controle de prioridade**

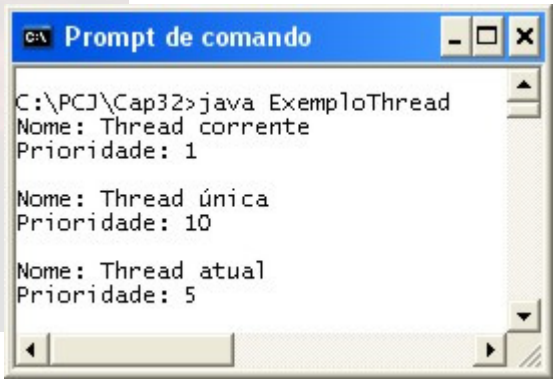
```
Thread th = Thread.currentThread( );
th.setPriority(10);
System.out.println( th.getPriority( ) );
th.setPriority(Thread.MAX_PRIORITY);
System.out.println( th.getPriority( ) );
```

- **Controle do nome**

```
Thread th = Thread.currentThread( );
th.setName("Linha corrente");
System.out.println( th.getName( ) );
```

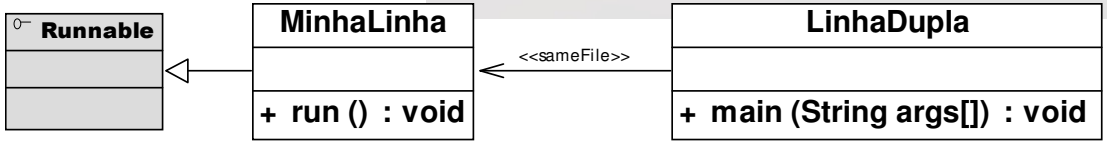
Linha de Execução Principal

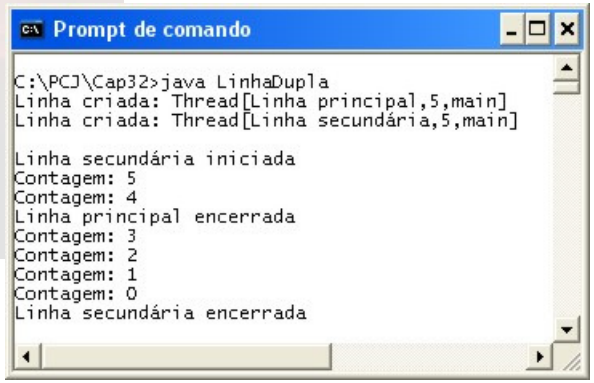
- **Código 32.1 – ExemploThread.java**
  - **Crie um aplicativo que recupere uma referência à linha de execução principal e realize as seguintes operações com ela:**
    - **Mude seu nome para "Thread corrente"**
    - **Mude sua prioridade para o nível mínimo**
    - **Imprima seu nome e prioridade**
    - **Realize uma pausa de 1 segundo**
    - **Mude seu nome para "Thread única"**
    - **Mude sua prioridade para o nível máximo**
    - **Imprima seu nome e prioridade**
    - **Realize uma pausa de 1 segundo**
    - **Mude seu nome para "Thread atual"**
    - **Mude sua prioridade para o nível normal**
    - **Imprima seu nome e prioridade**



Linha de Execução Secundária

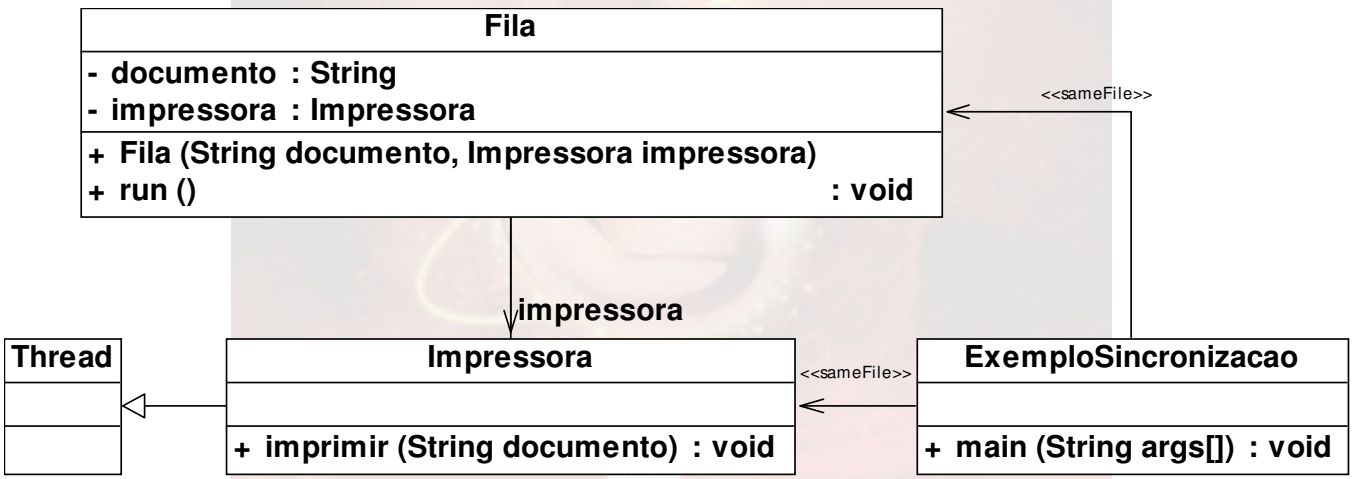
- **Código 32.2 – LinhaDupla.java**
  - **Classe MinhaLinha**
    - **run( ):**
      - **Imprimir: "Linha secundária iniciada"**
      - **Efetuar contagem: de 5 a 0**
      - **Imprimir: "Linha secundária encerrrada"**
  - **Classe LinhaDupla**
    - **main( ):**
      - **Recuperar referência à linha de execução principal**
      - **Alterar seu nome para "Linha principal" e a prioridade para 5**
      - **Imprimir representação textual da linha de execução principal**
      - **Criar linha de execução secundária utilizando a classe MinhaLinha**
      - **Alterar seu nome para "Linha secundária" e a prioridade para 5**
      - **Imprimir sua representação textual**
      - **Iniciar linha de execução secundária**
      - **Pausar a thread principal por 2 seg.**
      - **Imprimir: "Linha principal encerrada"**





Sincronização de Linhas de Execução

❑ Código 32.3 – ExemploSincronizacao.java

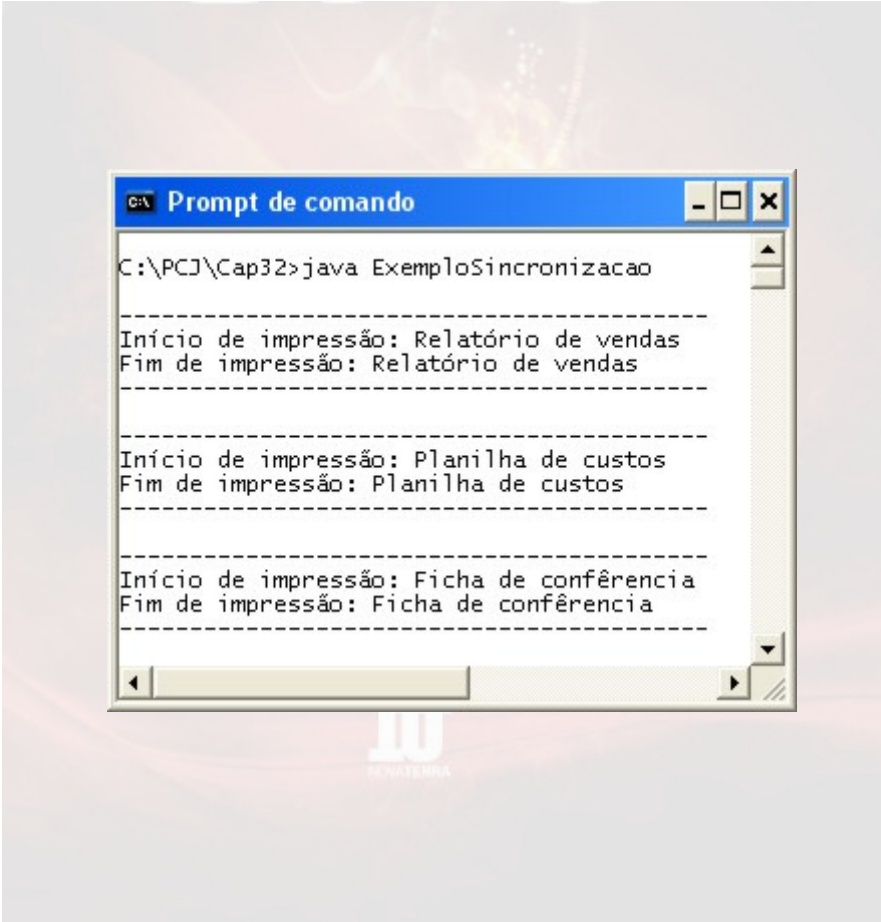


Sincronização de Linhas de Execução

- **Código 32.3 – ExemploSincronizacao.java**
  - **Classe Impressora**
    - **imprimir(String documento): método sincronizado**
      - **Imprimir: "Inicio de impressão: <documento>"**
      - **Realizar pausa de 3 segundos**
      - **Imprimir: "Inicio de impressão: <documento>"**
  - **Classe Fila**
    - **run( ): executar o método imprimir( ) da impressora**
  - **Classe ExemploSincronizacao**
    - **main( ):**
      - **Criar uma impressora (objeto da classe Impressora)**
      - **Vincular 3 documentos a esta impressora (objetos da classe Fila)**
      - **Ordenar a impressão dos 3 documentos (método start( ))**

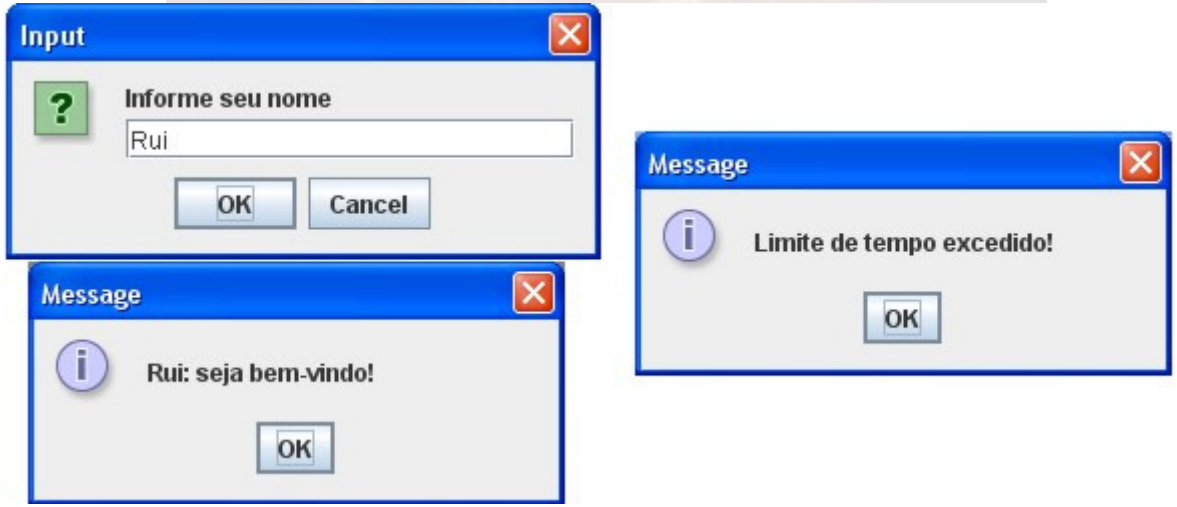
Sincronização de Linhas de Execução

- **Código 32.3 – ExemploSincronizacao.java**



Exercício 1

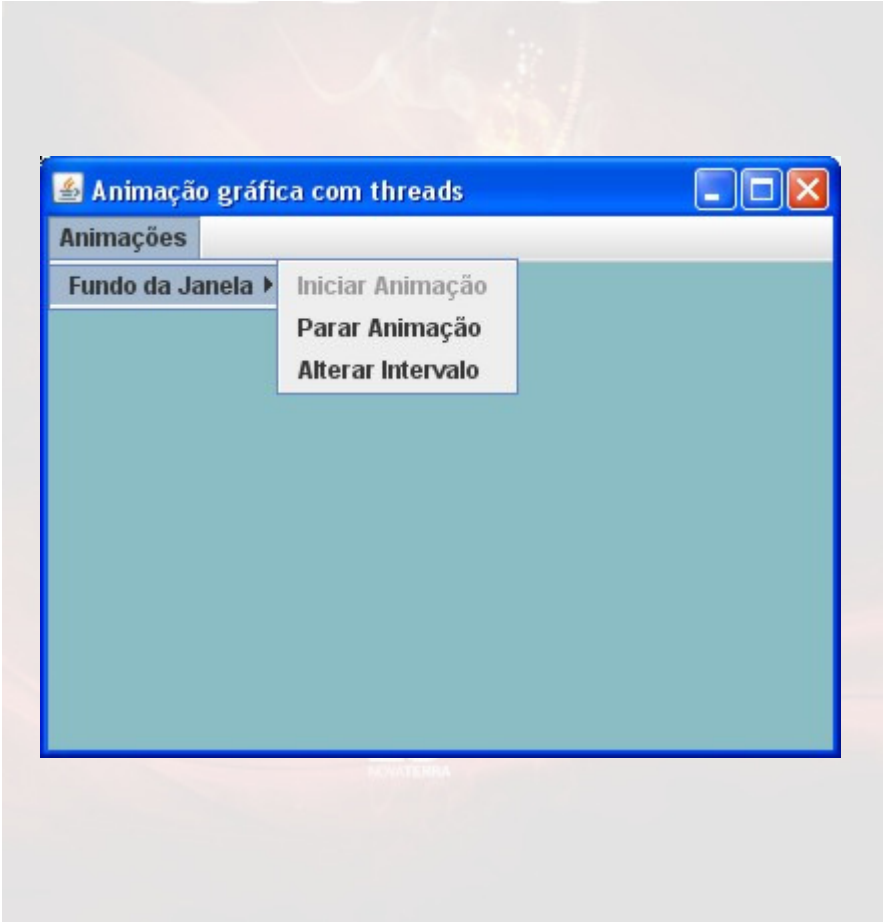
- Crie um novo aplicativo, chamado LimiteTempo, que capte o nome do usuário e exiba uma mensagem de boas-vindas.
  - A figura abaixo ilustra estes diálogos.
- Se o usuário demorar mais de dez segundos para informar seu nome e confirmar este dado, este aplicativo deve exibir uma mensagem que indique que o limite de tempo disponível foi excedido e ele deve ser encerrado.
  - Utilize uma linha de execução secundária para contar o tempo transcorrido a partir da exibição do diálogo de entrada no qual o nome do usuário deve ser informado.



Exercício 2

- Crie uma nova janela, chamada JFAnimacao, que possua uma barra de menus com as opções apresentadas pela figura do slide seguinte.
- O menu rotulado como “Fundo da Janela” deve ter três itens e eles devem permitir o controle de um efeito visual que pode ser ativado sobre o fundo da janela.
  - Este efeito consiste em trocar a cor de fundo da janela através de pequenas variações na cor atual e que pode ser feito por pequenos incrementos ou decrementos nas intensidades de vermelho, de verde e de azul que compõem a cor atual.
    - Cada nova cor aplicada ao fundo da janela deve ser similar à anterior, evitando mudanças bruscas.
  - O primeiro item deste menu servirá para iniciar a animação do fundo da janela, o segundo item servirá para interrompê-la e o terceiro item deverá apresentar um diálogo que permita configurar o intervalo de tempo (em milissegundos) que deverá haver entre uma mudança de cor e a mudança seguinte.
  - Quando o primeiro item estiver habilitado, o segundo deve permanecer desabilitado e vice-versa.

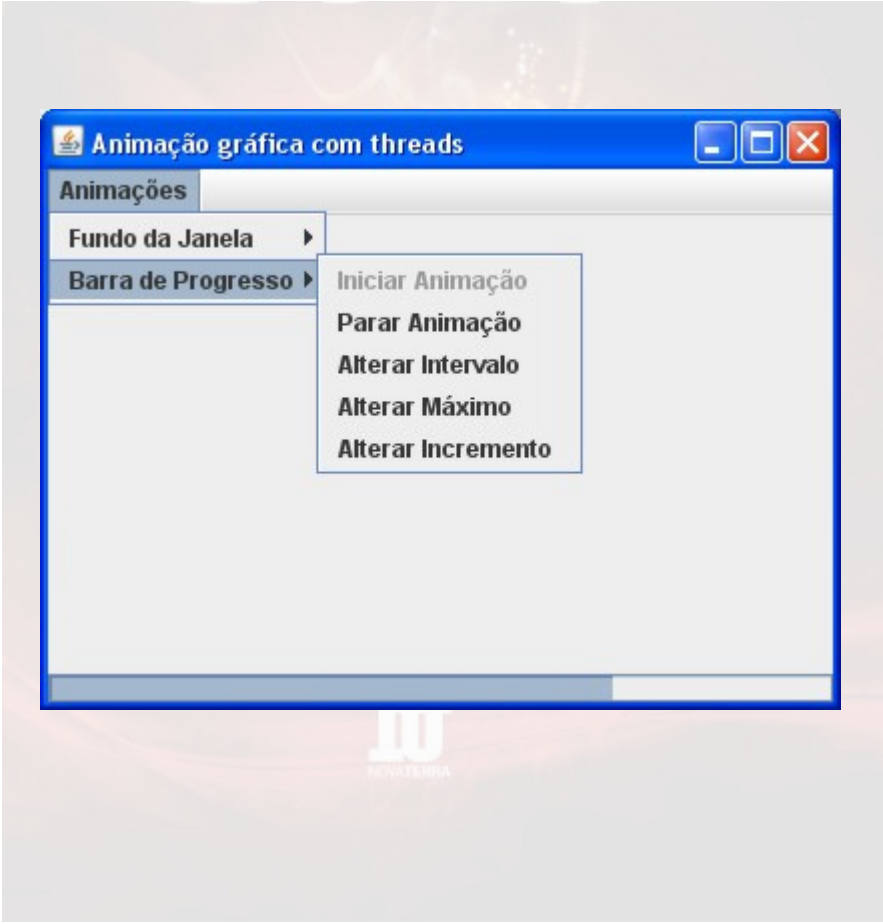
Exercício 2


Animação gráfica com threads
Animações
Fundo da Janela
Iniciar Animação
Parar Animação
Alterar Intervalo

Exercício 3

- Adicione um novo menu à janela que fora criada no exercício anterior.
  - Este menu deve ser rotulado como “Barra de Progresso” e deve ter os itens apresentados na figura do slide seguinte.
- Também adicione uma barra de progresso próximo à borda inferior desta janela.
  - Este componente pode ser criado utilizando-se a classe javax.swing.JProgressBar.
  - Você pode utilizar os métodos setMaximum( ) e getMaximum( ) desta classe para alterar e recuperar o valor máximo da barra de progresso, respectivamente.
  - Os métodos setValue( ) e getValue( ), por sua vez, podem ser empregados para alterar e recuperar o valor atual da mesma.
- Os cinco itens de menu adicionados devem permitir o controle de um efeito visual que pode ser ativado sobre a barra de progresso supracitada.
  - Este efeito consiste em atualizá-la a cada intervalo de tempo, utilizando o valor do incremento, até que ela atinja o valor máximo.
  - Os três últimos itens devem permitir a alteração do intervalo de tempo que deve ser transcorrido entre duas atualizações, do incremento a ser aplicado a cada vez e do valor máximo que a barra de progresso pode atingir.

Exercício 3


Animação gráfica com threads
Animações
Fundo da Janela
Barra de Progresso
Iniciar Animação
Parar Animação
Alterar Intervalo
Alterar Máximo
Alterar Incremento

Contato

Com o autor:

Rui Rossi dos Santos
E-mail: livros@ruirossi.pro.br
Web Site: http://www.ruirossi.pro.br

Com a editora:

Editora NovaTerra
Telefone: (21) 2218-5314
Web Site: http://www.editoranovaterra.com.br