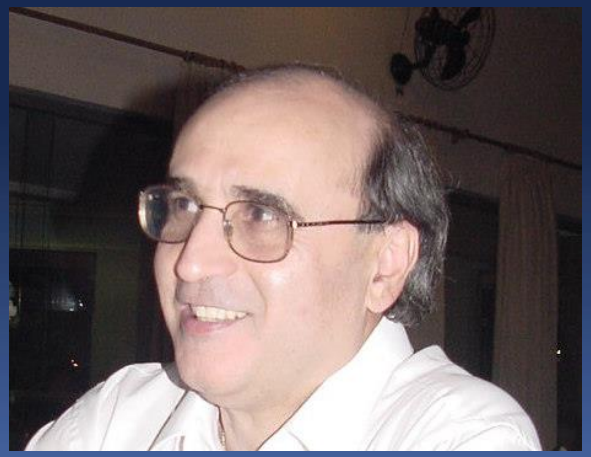
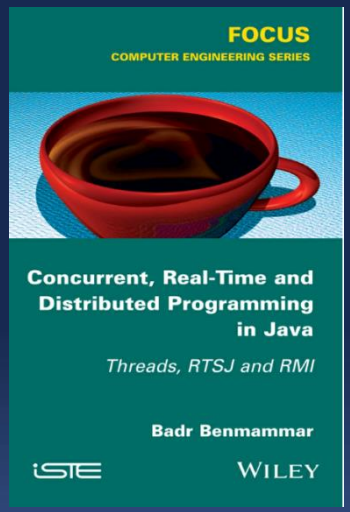
Programação Paralela e Concorrente

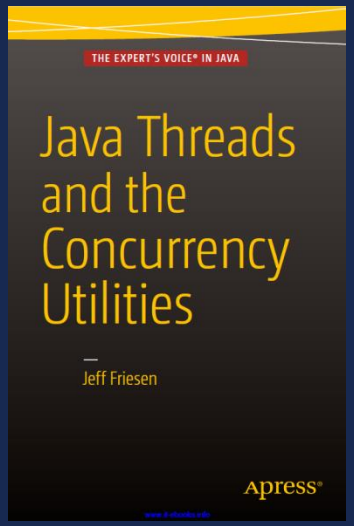
Unidade 7 – Sincronização de Threads
Outro exemplo

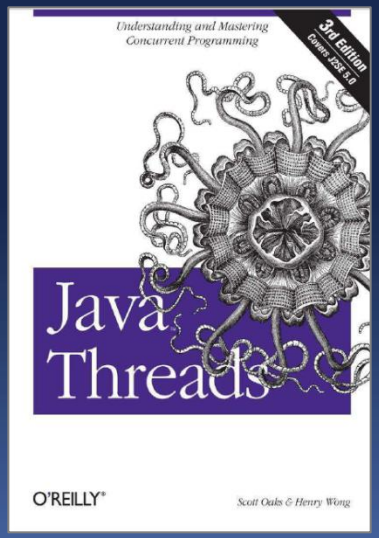


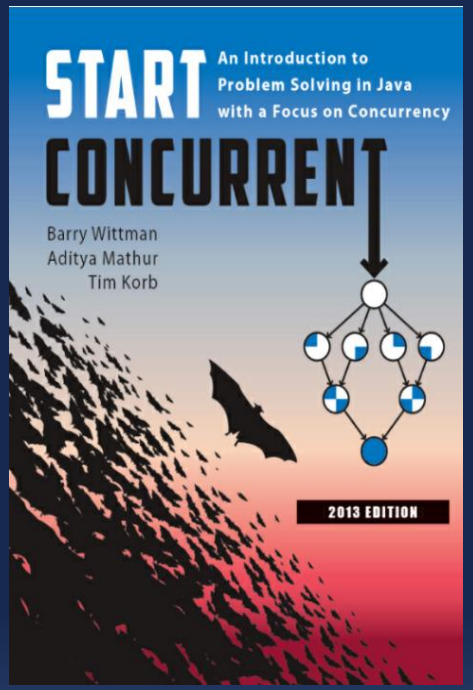
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

Bibliografia









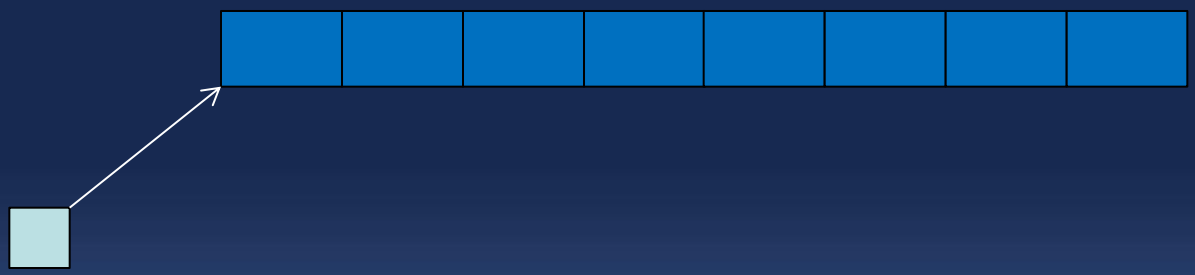
Introdução

- Na última unidade, vimos que quando dois ou mais threads acessam um mesmo objeto ao mesmo tempo, podem ocorrer anomalias no processamento;
- Nesta unidade, veremos mais um exemplo de uma aplicação no qual vários threads estão acessando um mesmo recurso;
- Vamos construir uma aplicação no qual diversos threads estarão acessando uma lista de dados comum à todos eles.

Aplicação

Classe Lista

- Vamos criar uma classe chamada Lista que representará o objeto que estará sendo compartilhado por diversos threads.



Classe Lista

- Na classe **Lista** iremos definir um array de 5000 elementos do tipo String;
- O método **insereString()** recebe um objeto do tipo String, adiciona-o na lista numa posição i e também incrementa essa posição i;
- O método **tamanho()** retorna o tamanho total da Lista;
- O método **recuperaString()** recebe uma posição i e retorna o String armazenado nessa posição.

Classe Lista

```
package br.uscs;

public class Lista {

    //objeto lista com 5000 elementos será compartilhado entre 10  threads
    private String[] listaString = new String[5000];
    private int indice = 0;


    public  void insereString(String elemento) {
        this.listaString[indice] = elemento;
        this.indice++;
    }

    public int tamanho() {
        return this.listaString.length;
    }

    public String recuperaString(int posicao) {
        return this.listaString[posicao];
    }
}
```

Definição da Tarefa a ser executada

- Definiremos uma tarefa chamada **TaskAdicionaString** que será executada pelos **threads**, caracterizando dessa forma uma **Programação Concorrente**.
- A tarefa **TaskAdicionaString** será portanto do tipo **Runnable** e será responsável por inserir Strings na lista;
- Criaremos assim uma classe chamada **TaskAdicionaString** que deverá portanto implementar a interface Runnable;
- Nessa aplicação, iremos considerar que **10 threads** irão executar essa task em paralelo .

Classe TaskAdicionaString

```java
package br.uscs;

public class TaskAdicionaString implements Runnable{

    private Lista lista;
    private int numeroDoThread;

    // Construtor
    public TaskAdicionaString(Lista lista, int numeroDoThread) {
        this.lista = lista;
        this.numeroDoThread = numeroDoThread;
    }
```

Classe TaskAdicionaString

```
@Override
public  void run() {
    // Temos 10 threads para gravar 5000 elementos no array
    // cada thread irá, portanto, gravar 500 elementos no array
    // teremos ao final da aplicação todo o array preenchido com 5000 elementos

    for(int i=0; i < 500; i++) {
        String textoGerado = geraString();
        lista.insereString("Thread " + numeroDoThread + " gravou ===> " + textoGerado);
    }
}
```

Classe TaskAdicionaString

```
    public static String geraString() {

        String textoGerado = " ";
        int indice;
        String alfabeto = new String("!@#$%^&*<>abcdefghijklmnopqrstuvxyzwABCDEFGHIJKLMNOPQRSTUVXZW0123456789");

        for (int i=0 ; i<20; i++) {
            indice = (int) (70.0 * Math.random());
            String sub = alfabeto.substring(indice, indice+1);
            textoGerado = textoGerado + sub;
        }
        return textoGerado;
    }
}
```

Classe Principal

- Nesta classe criaremos threads que irão manipular o objeto Lista da classe Lista definida anteriormente;
- Vamos criar nessa classe 10 threads por meio de um laço for;
- Dentro desse laço, serão criados os threads, cada qual processando a tarefa TaskAdicionaString(), recebendo a lista compartilhada como argumento.

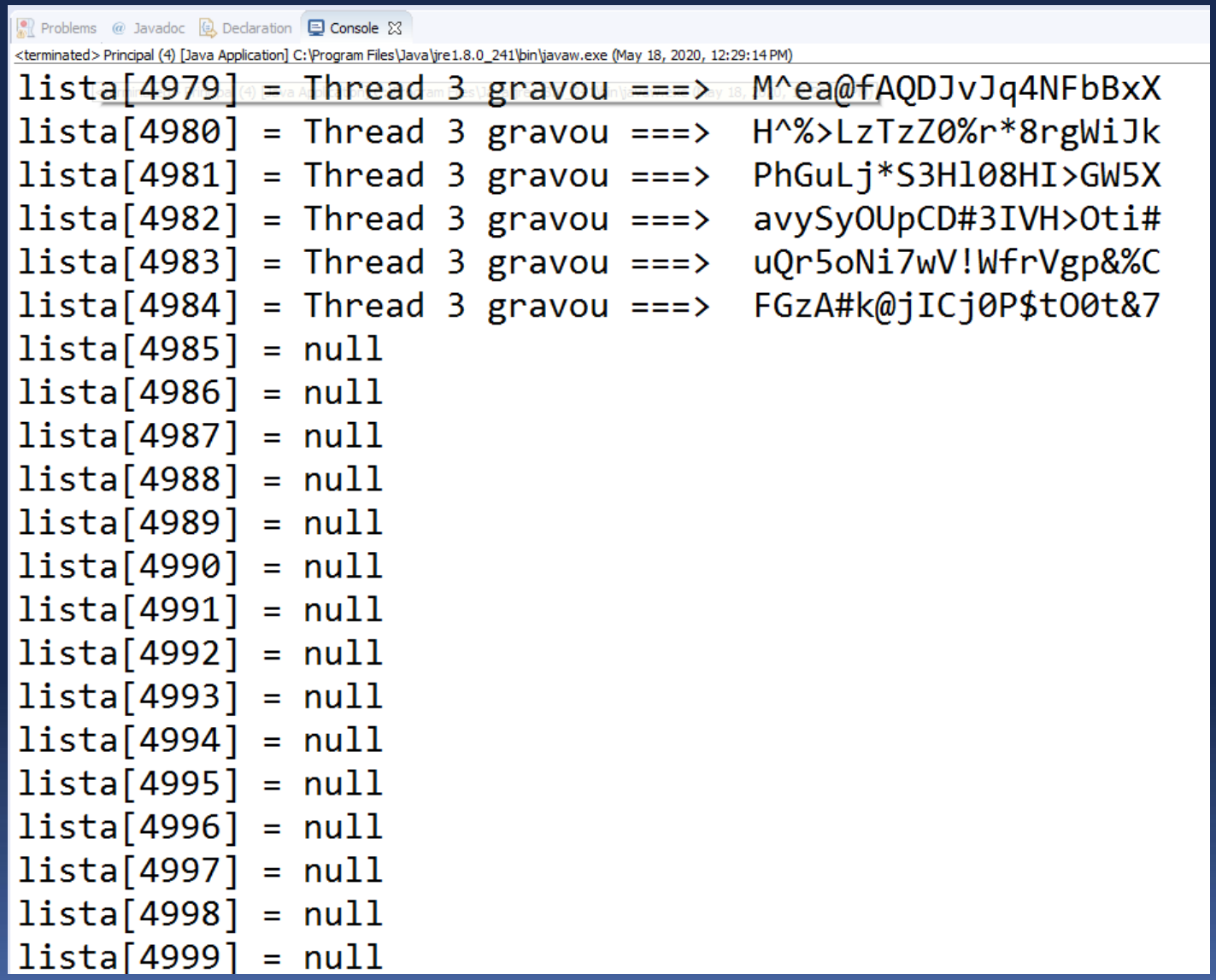
Classe Principal

```
package br.uscs;

public class Principal {

    public static void main(String[] args) {

        Lista lista =  new Lista();  //objeto lista será compartilhado

        // A aplicação irá operar com 10 threads

        for (int i=0; i<10; i++) {

            TaskAdicionaString task = new TaskAdicionaString(lista,i);
            Thread thread = new Thread(task, "Thread T" + i);
            thread.start();
        }
```

Classe Principal

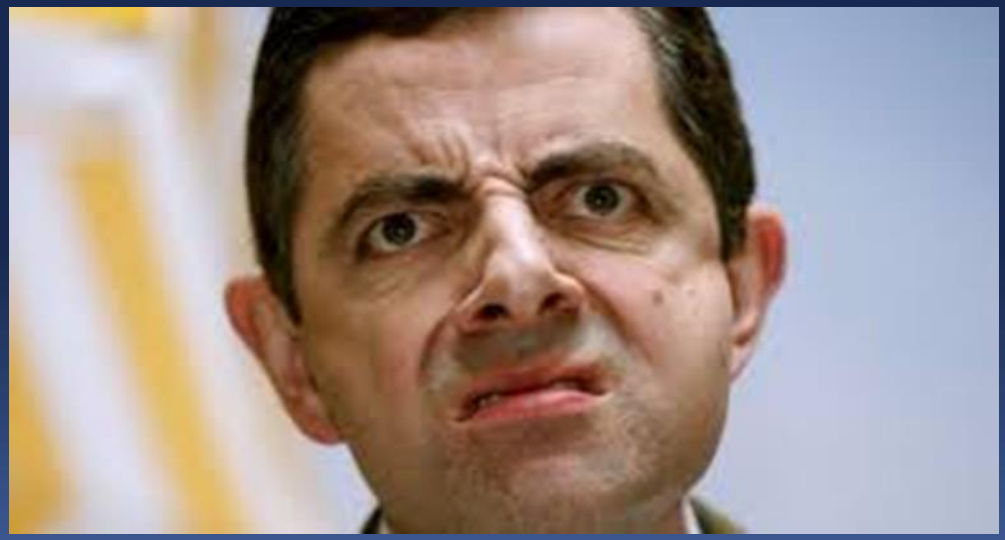
```
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        //listando os valores armazenados nas 5000 posições do objeto lista
        for (int i=0; i<5000; i++) {
            System.out.println("lista[" + i +"] = " + lista.recuperaString(i));
        }
    }
}
```

Executando a aplicação



Vé ... Mas, algumas posições do array estão com valores nulos ????

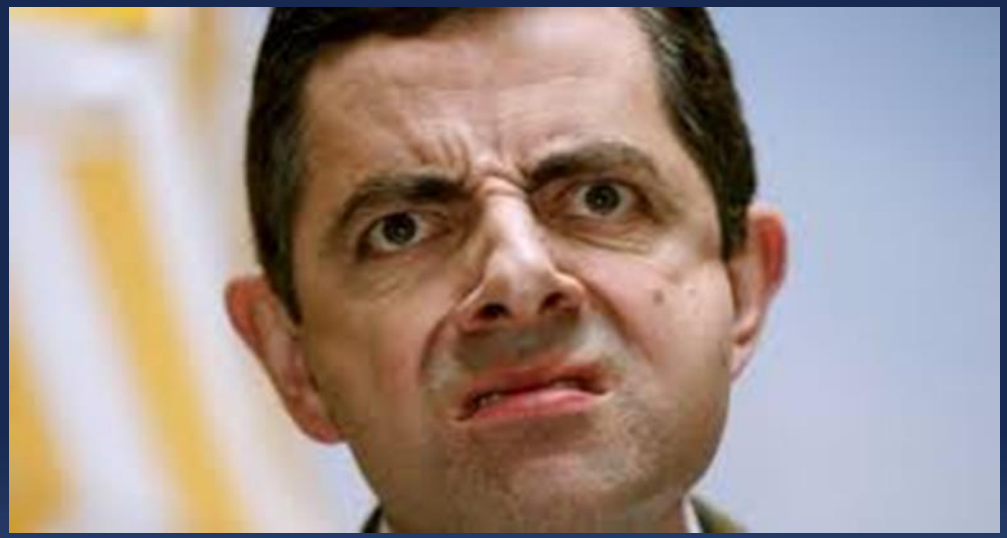
Por que?



Erro na execução ???

- Ocorreram erros pois vários threads estão acessando simultaneamente o mesmo objeto;
- Ou seja, um thread está tentando acessar uma posição que ainda não foi incrementada depois do acesso do thread anterior, e os dois acabam utilizando o mesmo índice, de modo que o novo precisa sobrescrevê-lo.
- Porém, ambos os threads acabam incrementando a posição, gerando uma posição nula (null) no processo.

Como resolver ???



synchronized

```
package br.uscs;

public class Lista {

    //objeto lista com 5000 elementos será compartilhado entre 10  threads
    private String[] listaString = new String[5000];
    private int indice = 0;


    //synchronized ==>  essa função está sendo chamada simultaneamente pelos threads
    public  synchronized  void insereString(String elemento) {
        this.listaString[indice] = elemento;
        this.indice++;
    }

    public int tamanho() {
        return this.listaString.length;
    }

    public String recuperaString(int posicao) {
        return this.listaString[posicao];
    }
}
```

synchronized

```
lista[4986] = Thread 7 gravou ===>  ISl$U%fhq&UGX@ZOjiNj
lista[4987] = Thread 7 gravou ===>  GAiJnXhUzzpj>BAL$RQ5
lista[4988] = Thread 7 gravou ===>  C7KDlxL$HblkS%QW5KqQ
lista[4989] = Thread 7 gravou ===>  @*2d#%Jx<x@rIbWUoC>%
lista[4990] = Thread 7 gravou ===>  ^!^!^Ubz<HOwFvCUJbGg
lista[4991] = Thread 7 gravou ===>  lohas4EWJn<EK<RgJBlq
lista[4992] = Thread 7 gravou ===>  fBGtTqA6sS>8Zq>K5e7Q
lista[4993] = Thread 7 gravou ===>  U!D#M3pQT6cpTR1u825M
lista[4994] = Thread 7 gravou ===>  HJWhSyDxJ!G%xZyAbiTe
lista[4995] = Thread 7 gravou ===>  wAmibj81f2F5mnRG80LH
lista[4996] = Thread 7 gravou ===>  zp&rOzq%e@4mbuC0l<hA
lista[4997] = Thread 7 gravou ===>  &ZVzywPACTbwvj>2TlQa
lista[4998] = Thread 7 gravou ===>  462ov2Tr0^xL^6T8qotv
lista[4999] = Thread 7 gravou ===>  jnOQi^FiVc#Bw0zQAMh2
```