

## A programação concorrente

Tarefa:

### Serviço a ser executado

Define um conjunto de instruções a serem executadas de forma seqüencial.

Sincronização:

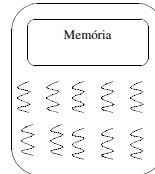
### Troca de informações coordenada entre duas tarefas

Mecanismo que permite controlar o compartilhamento de dados entre tarefas.

### Comunicação

## Threads em Java

Multithreading: conceito



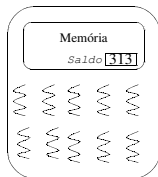
Dentro de um processo, instruções são executadas por vários fluxos concorrentes.

Eventualmente alguns destes fluxos de execução podem estar bloqueados

Em operações de sincronização (E/S, comunicação, ...)

## Threads em Java

Multithreading: compartilhamento de memória

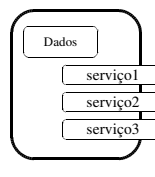


### Problema

Thread A	Thread B
<b>Deposita:</b> saldo = saldo + 1	<b>Saca:</b> saldo = saldo - 1
A1: mov AX, [Saldo] A2: add AX, 1 A3: mov [saldo], AX	B1: mov AX, [Saldo] B2: dec AX, 1 B3: mov [saldo], AX

## Threads em Java

Multithreading: compartilhamento de memória



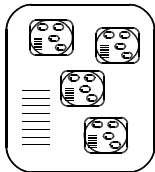
### Solução: Monitor

É uma estrutura de sincronização que permite associar um conjunto de serviços à uma área de dados.

É garantido que apenas um dos serviços está ativo em um instante de tempo.

## Concorrência em Java

Objetos são entidades autônomas – portanto podem possuir fluxo de execução próprio



### Programação OO:

Um programa um conjunto de objetos interagindo e cooperando entre si.

Um objeto pode possuir um fluxo de execução próprio como um programa qualquer.

## Concorrência em Java

### Threads

Uma thread é um fluxo de execução dentro de um programa

A classe *abstrata* Thread implementa o mecanismo de suporte a execução concorrente.

```

abstract class Thread {
    void start() { ... }
    virtual void run();
    void join() { ... }
}

public class NOME extends Thread {
    void run() {
        faz alguma coisa
    }
}
    
```

## Concorrência em Java

### Exemplo:

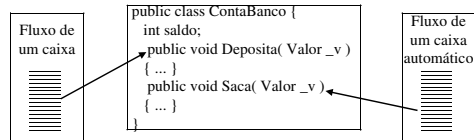
```
public class ImpStr extends Threads {
    private String str;
    private int n;
    public ImpStr( String _s, int _n ) {
        str = new String( _s );
        n = _n;
    }
    public void run() {
        for( int i = 0 ; i < n ; i++ )
            System.out.println(str);
    }
}

public class Principal {
    public static void main( String[] args ) {
        String s = new String( "Gerson" );
        ImpStr imp = new ImpStr( s, 10 );
        imp.start();
        ....
        imp.join();
    }
}
```

## Concorrência em Java

### Sincronização entre objetos

Como cada objeto tendo seu fluxo de execução próprio, podem ocorrer situações onde **um objeto recebe duas ou mais invocações simultâneas**



## Concorrência em Java

### Sincronização entre objetos

Como cada objeto tendo seu fluxo de execução próprio, podem ocorrer situações onde **um objeto recebe duas ou mais invocações simultâneas**

```
public class ContaBanco {
    private int Saldo;
    public synchronized void Deposita( int _v )
    { Saldo = Saldo + _v; }
    public synchronized void Sacar( int _v )
    { Saldo = Saldo - _v; }
}
```

## Concorrência em Java

**Exemplo:** encontrar o maior valor em um vetor de inteiros positivos

```
public class Maior {
    private int maior;

    public Maior() { maior = 0; }

    public synchronized void setMaior( int _m )
    { if( _m > maior ) maior = m; }

    public synchronized int getMaior()
    { return maior; }
}
```

## Concorrência em Java

**Exemplo:** encontrar o maior valor em um vetor de inteiros positivos

```
public class Acha extends Thread {
    private Maior maior;
    private VectInt v;
    private int posic, nb;

    public Acha( Maior _m, VectInt _v, int _p, int _nb )
    { maior = _m; v = _v; posic = _p; int nb = _nb; }

    public void run() { // serviço que procura o maior
        ... // no intervalo [posic, posic+nb)
        maior.setMaior( aux );
    }
}
```

## Concorrência em Java

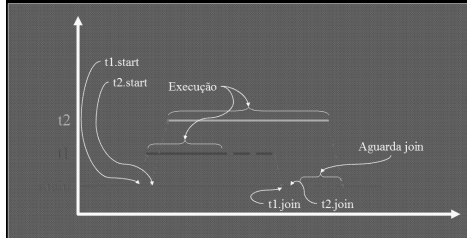
**Exemplo:** encontrar o maior valor em um vetor de inteiros positivos

```
... main ... {
    Acha t1, t2;
    VectInt vet(100);
    Maior m;

    t1 = new Acha(vet, m, 0, 50);
    t2 = new Acha(vet, m, 50, 50);
    t1.start();
    t2.start();
    ...
    t1.join();
    t2.join();
    System.out.println("Maior = " + m.getMaior());
}
```

## Concorrência em Java

**Exemplo:** encontrar o maior valor em um vetor de inteiros positivos



## Concorrência em Java

**Exercício:** calcular a integral de uma curva através do método dos trapézios

**Solução:**

- uma classe para a área total  
`class Area`
- uma classe para o cálculo  
`NewtonCotes extends Threads`
- um m todo dividindo a área total entre diversas threads

