Tratamento de Exceções, Multithreads e arquivos (em Java)

Programação Orientada a Objetos

educação a distância
UFSCar
virtual

Programação Orientada a Objetos

Nesta unidade vamos ver os últimos assuntos de interesse em java.

O primeiro deles, bem simples, é o ***tratamento de exceções*** (você pode pesquisar como é o tratamento de exceções em C++).

Programação Orientada a Objetos

Em tratamento de exceções, temos um bloco try:

```
try
{
   ...// instruções que podem lançar uma exceção
}
Catch (tipo)
{
   ...// instruções para tratar uma exceção
}
```

Programação Orientada a Objetos

Catch "pega" uma exceção disparada por **throw**.

Assim, para disparar uma exceção, precisamos fazer uso da instrução **throw**.

Programação Orientada a Objetos

Se o bloco **try** for executado sem problemas, então nenhum tratador de exceções é executado.

Podem existir vários blocos **catch** em um bloco **try**.

Pode existir um bloco **finally** (colocado logo após o último **catch**) que será executado independentemente de uma exceção ser ou não disparada.

Programação Orientada a Objetos

Exemplo (esquemático):

```
try
{   ...
        resultado = divisao(p,q);
    ...
}
catch (DivisaoPorZero e)
{
    // mostrar mensagem de erro
}
```

Programação Orientada a Objetos

Exemplo (esquemático):

```
public double divisao(int x, int y) throws DivisaoPorZero
{
    if (y == 0)
        throw new DivisaoPorZero();

    return (double) x/y;
}
```

Programação Orientada a Objetos

Exemplo (completo)

```
public class DivisaoPorZero extends Exception
{
    public DivisaoPorZero()
    {
        JOptionPane.showMessageDialog(null," Erro! Tentativa de divisão
por zero", "teste",  JOptionPane.PLAIN_MESSAGE);
    }
}
```

Programação Orientada a Objetos

Exemplo (código main)

```
TesteDivisaoPorZero x = new TesteDivisaoPorZero();
try
{
  System.out.println(x.divisao());
}
catch(DivisaoPorZero z)
{
  System.out.println("erro");
}
```

Programação Orientada a Objetos

Exemplo (método divisão)

```
public double divisao() throws DivisaoPorZero
{
    if (n2 == 0)
       throw new DivisaoPorZero();
    resultado = (double) n1/n2;
    return resultado;
}
```

Programação Orientada a Objetos

CONSTRUTOR

```
public TesteDivisaoPorZero()
{
    n1 = 10;
    n2 = 0;
    resultado = 0;
}
```

Programação Orientada a Objetos

A cláusula throws lista as exceções que podem ser disparadas pelo método:

```
Tiporet  nomeMétodo(parametros) throws exceção1, exceção2, ...
```

Programação Orientada a Objetos

Conclusão: Este conceito (tratamento de exceções) é bem simples de se entender. É preciso consultar outras fontes bibliográficas para consolidar seu conhecimento e verificar outros casos (como relançamento de uma exceção).

Programação Orientada a Objetos

Multithreading

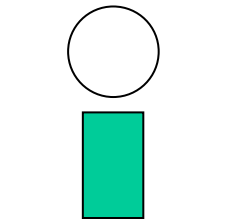
Programação Orientada a Objetos

Threads ajudam a produzir ações simultâneas.

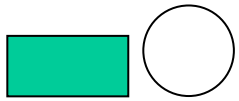
Imagine que tenhamos 2 trabalhadores em um escritório. Os dois estão meio sonolentos e, vez ou outra, dormem um pouco.

Com threads fica fácil simular este comportamento (independente e paralelo) dos trabalhadores.

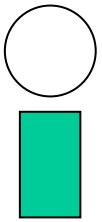
Programação Orientada a Objetos



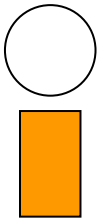
acordado



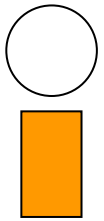
dormindo



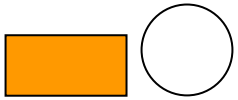
acordado



acordado



acordado



dormindo

Programação Orientada a Objetos

Como o computador permite que os processos trabalhem em paralelo?

Na verdade cada processo recebe uma fração de tempo para executar sua tarefa. Quando o tempo termina, o processo é colocado em estado de espera passando a vez para outro processo...

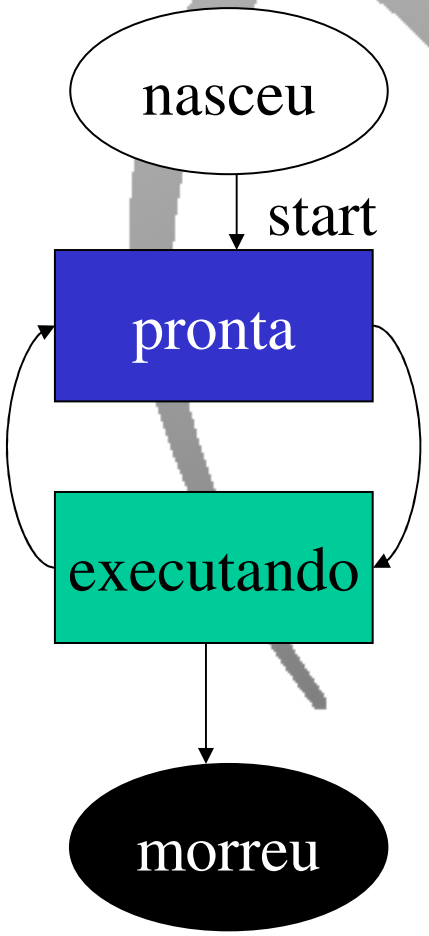
Programação Orientada a Objetos

Os processos com mesma prioridade dividem igualmente o recurso CPU em um rodízio chamado round-robin.

Depois que o último processo utilizou o recurso CPU, o primeiro da fila volta a ter direito de utilizar mais um pouco de CPU.

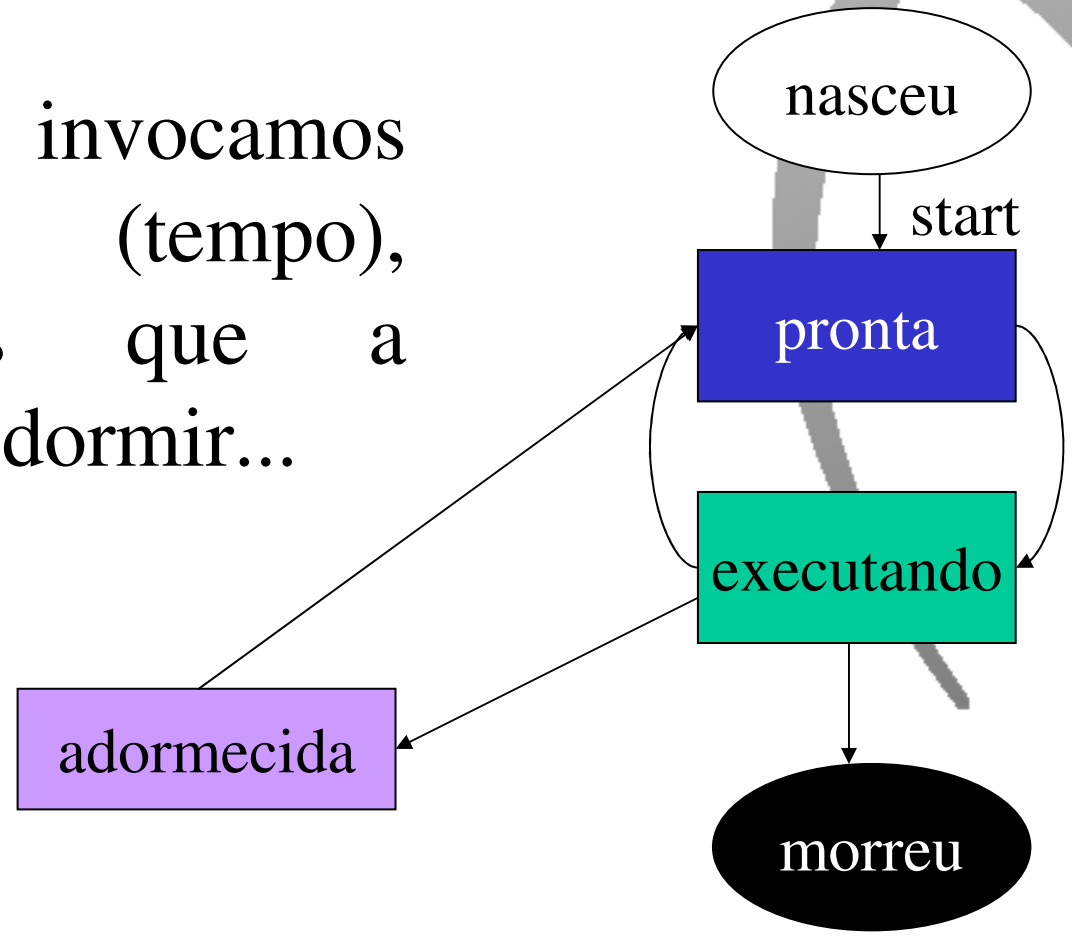
Programação Orientada a Objetos

Ciclo de vida de uma thread. Ela nasce com o comando start e é executada através de seu método run.



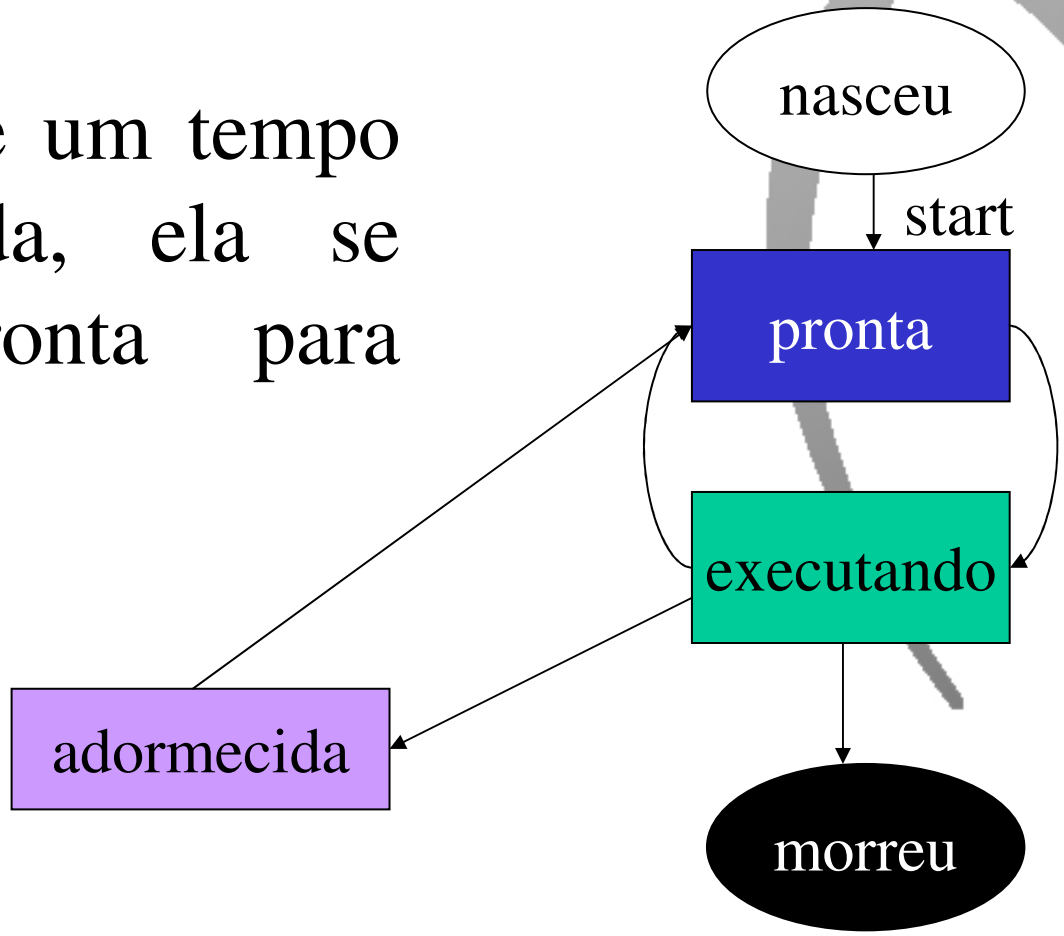
Programação Orientada a Objetos

Quando invocamos sleep (tempo), indicamos que a thread irá dormir...



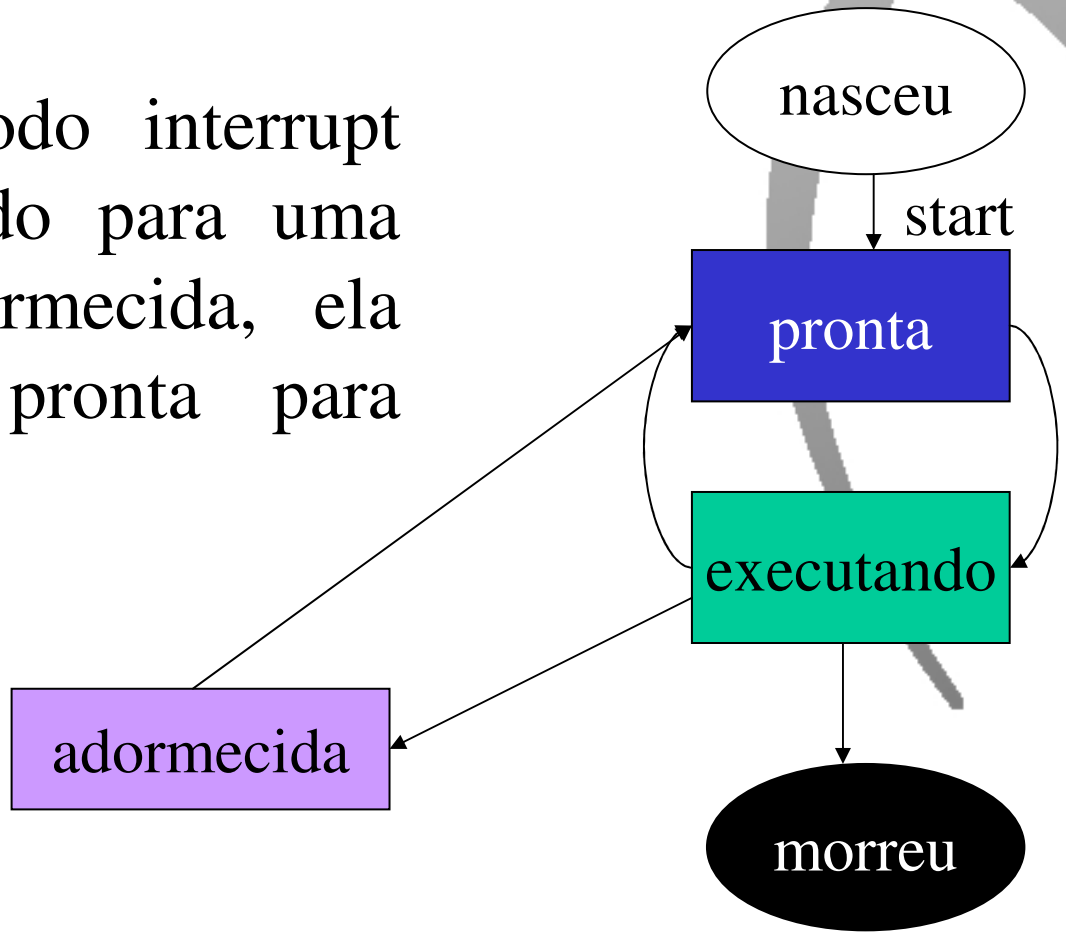
Programação Orientada a Objetos

Depois de um tempo adormecida, ela se torna pronta para executar...



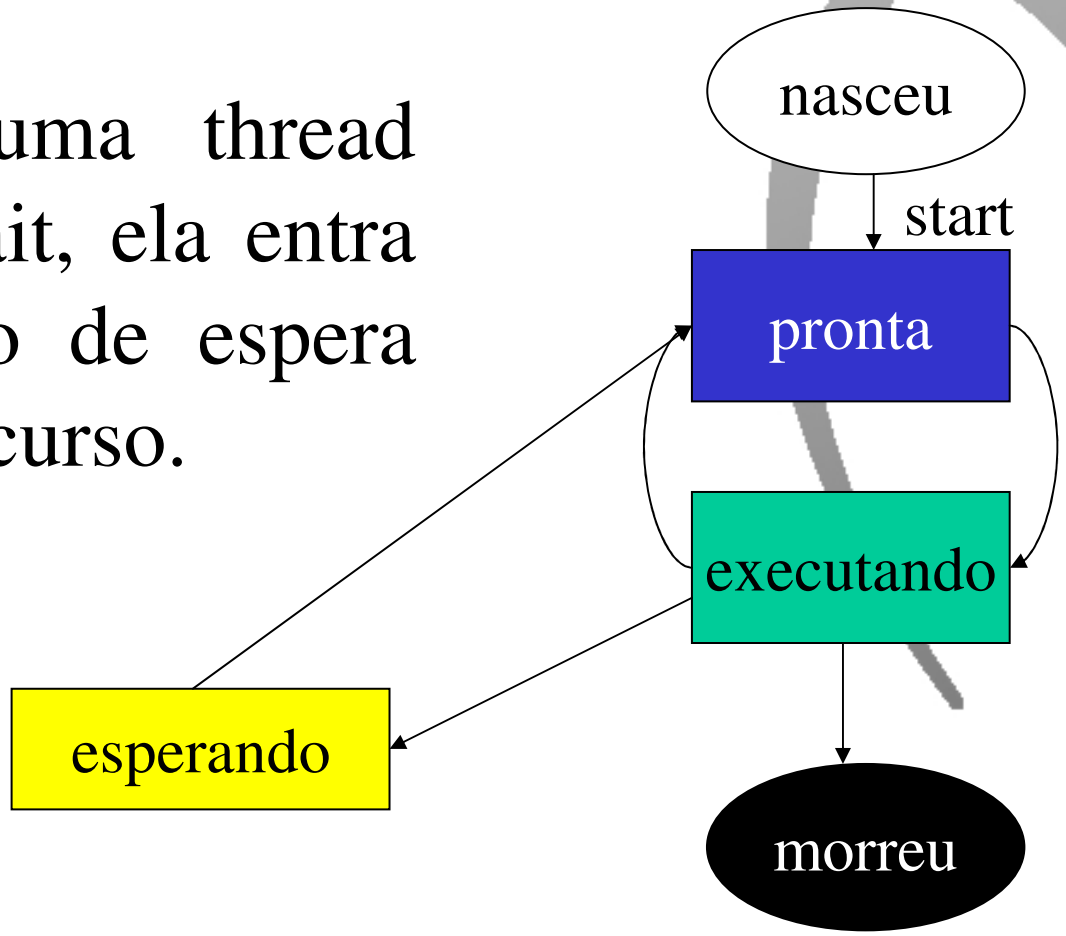
Programação Orientada a Objetos

Se o método interrupt for chamado para uma thread adormecida, ela se torna pronta para executar.



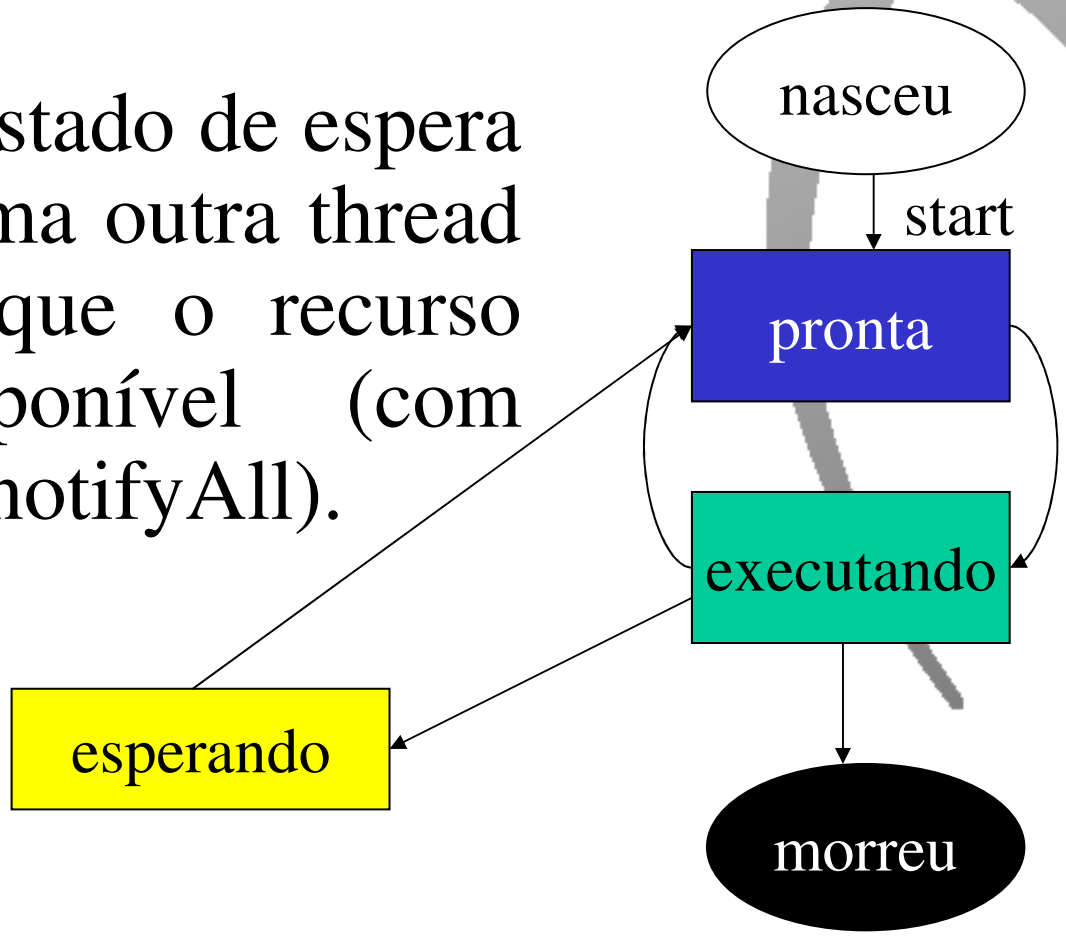
Programação Orientada a Objetos

Quando uma thread chama wait, ela entra em estado de espera por um recurso.



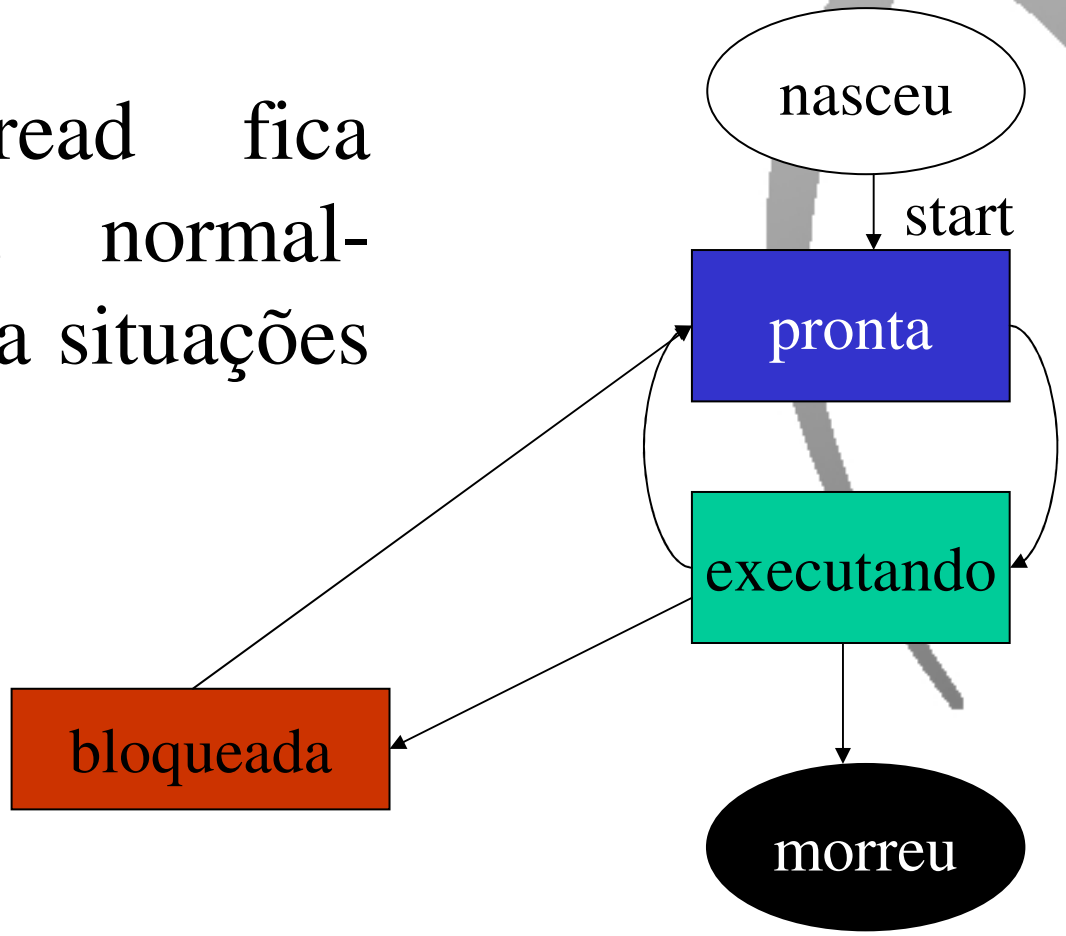
Programação Orientada a Objetos

Sairá do estado de espera quando uma outra thread notificar que o recurso está disponível (com notify ou notifyAll).



Programação Orientada a Objetos

Uma thread fica bloqueada normal-mente para situações de E/S.



Programação Orientada a Objetos

Um exemplo bem simples:

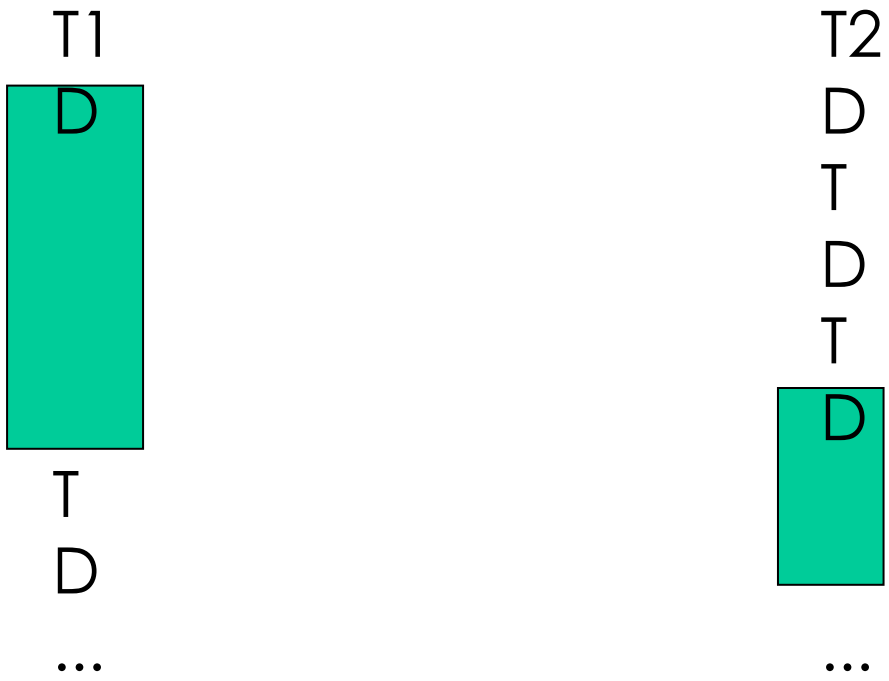
```
public class testeThread
{
      public static void main(String args[]) {
      processo T1,T2;
       T1 = new processo(“T1”);
       T2 = new processo(“T2”);
       T1.start(); // inicia a thread
       T2.start(); // inicia a outra thread
    }
}
```

Programação Orientada a Objetos

```
class processo extends Thread
{
    private int tempo;
    public processo (String nome)
    {
        super (nome);
        tempo = (int) (Math.random() * 300);
    }
    public void run ( )
    {
      int i=0;
      while (i < 10)
      {
            System.out.println(getName( )+ "dormindo");
            try{Thread.sleep(tempo);}
            catch (InterruptedException x){ System.out.println("erro");}
           System.out.println(getName( )+ "trabalhando");
           i++;
      }
    }
}
```

Programação Orientada a Objetos

Execução:



Programação Orientada a Objetos

Conclusões: Threads é um assunto bastante rico. Você está tendo apenas um aperitivo para saber do que se trata.

OBS.: POO objetiva – primordialmente – ensinar orientação a objetos.

Programação Orientada a Objetos

Arquivos (C++)

Programação Orientada a Objetos

Às vezes desejamos que as informações permaneçam armazenadas no computador quando o desligamos.

Para fazer isso, precisamos armazenar os dados em arquivos.

Quando armazenamos os dados em arquivos, chamamo-os de dados persistentes.

Programação Orientada a Objetos

marcador de fim de arquivo

Programação Orientada a Objetos

Os arquivos podem ser de acesso seqüêncial ou de acesso aleatório (randômico).

Nos arquivos de acesso seqüêncial, a informação na posição 100 só pode ser acessada após 99 leituras de dados.

Já nos arquivos de acesso aleatório, a informação na posição 100 seria obtida através de um único acesso.

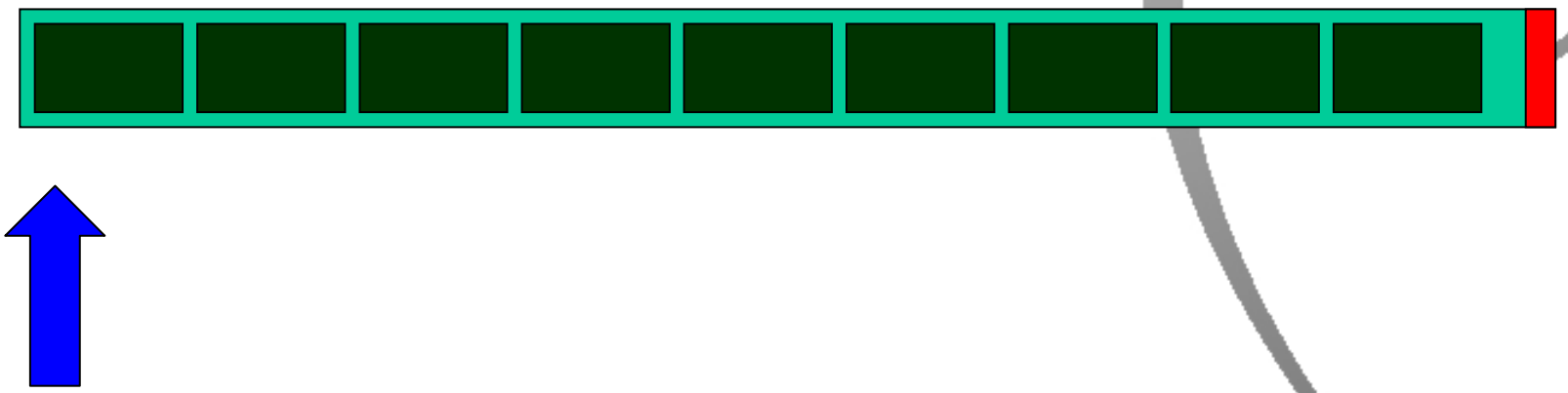
Programação Orientada a Objetos

Para acessar a informação em uma determinada posição do arquivo de acesso aleatório, é preciso posicionar o ponteiro do arquivo.

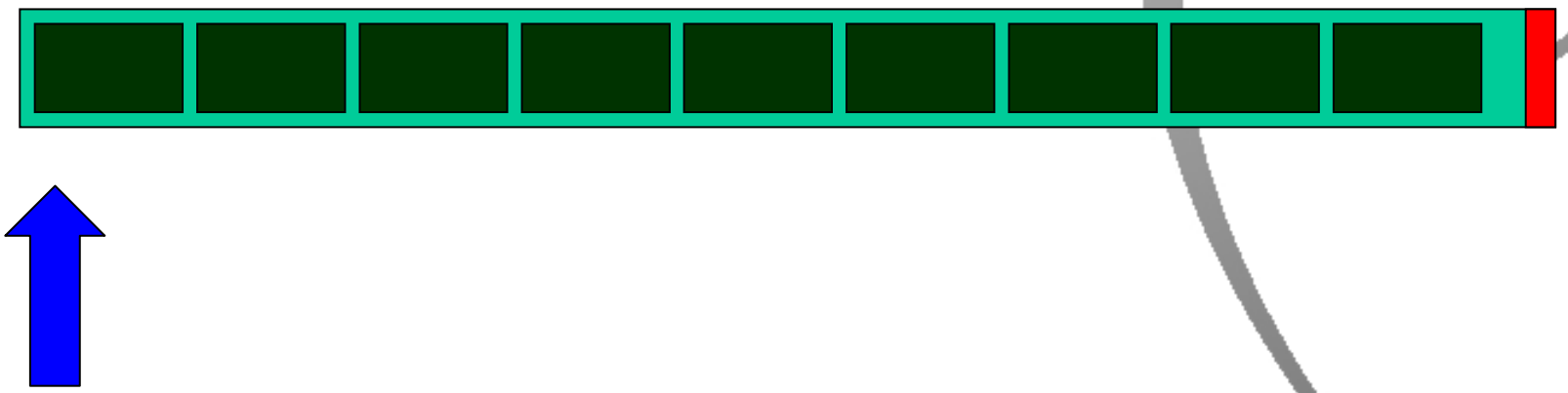
Isso é feito com o método seekg (seek get) ou seekp (seek put).

A abertura de arquivos normalmente coloca o ponteiro do arquivo na primeira posição.

Programação Orientada a Objetos



Programação Orientada a Objetos



Programação Orientada a Objetos

Um arquivo de acesso aleatório pode ser consultado de forma seqüêncial (um registro após o outro).

Um arquivo de acesso seqüêncial não pode ser consultado de forma aleatória!

Programação Orientada a Objetos

```
#include <fstream>  // file stream
 using std::ofstream; //output file stream
...
 ofstream arq("nome", ios::out); // saida
 if (!arq) {
       //erro
       exit(1);
 }
...
```

ios::app	→ append (final do arquivo)
ios::ate	→ abre como escrita posiciona no final do arquivo.
ios::in	→ abre como entrada
ios::binary	→ abre como binário

Programação Orientada a Objetos

```
#include <fstream>  // file stream
 using std::ofstream; //output file stream
...
 ofstream arq("nome", ios::out); // saida
 if (!arq) {
      //erro
      exit(1);
 }
...
```

```
ifstream arq("nome", ios::in); // entrada
```

Programação Orientada a Objetos

```
while (cin>> nome >> nota)
{
      arq << nome << “  “ << nota << endl;
 }
 ...
```

Saida de dados para o arquivo. Leitura do teclado.

Programação Orientada a Objetos

```
while (arq>> nome >> nota)
{
      cout << nome << “  “ << nota << endl;
 }
 ...
```

Saida de dados para o monitor. Leitura do arquivo.

Programação Orientada a Objetos

Para criar um arquivo de acesso aleatório é preciso abri-lo como binary e utilizar o comando write da seguinte forma:

arq.write (reinterpret_cast <const char*> (&n), sizeof(n));

Programação Orientada a Objetos

Para criar um arquivo de acesso aleatório é preciso abri-lo como binary.

Utilize o comando write para escrever informações no arquivo e o comando read para ler:

```
arq.write (reinterpret_cast <const char*> (&n), sizeof(n));
arq.read (reinterpret_cast <const char*> (&n), sizeof(n));
```

Programação Orientada a Objetos

Eis um loop de leitura de dados de forma seqüêncial de um arquivo de acesso aleatório:

```
REGISTRO reg;

while (arq && !arq.eof())
        arq.read(reinterpret_cast <const char *> (&reg), sizeof(reg));
```

Programação Orientada a Objetos

Para a leitura de uma informação em um determinado registro, basta utilizar seekg:

arq.seekg(pos*sizeof(reg));

O mesmo se aplica para o ponteiro de escrita (seekp).

Programação Orientada a Objetos

Trabalhar com arquivos também não é algo complicado. É preciso prática para compreender os comandos/métodos.

Estude primeiramente como o processo de leitura e escrita ocorre em C++ e depois pesquise como acontece em Java. Isso fará com que você consolide seu aprendizado.

Programação Orientada a Objetos

FIM