UTFPR – WebConf 02 – Professor Arthur Pereira Gregório.

Aluno: Elvir Moraes

WEBCONF 02

TAREFA NO CASO DA NÃO PARTICIPAÇÃO DA WEB CONFERÊNCIA:

1) Assistir a gravação da Web Conferência 2 e fazer um relato de forma textual dos conteúdos e atividades ministradas durante a conferência. Entregar um documento PDF.

THREADS

Um breve histórico de threads;

Nos computadores antigos com Windows 98, tinha o Winamp, um software de músicas, quando tocava, se mexesse no mouse a máquina poderia travar, que confundia um erro no sistema operacional com hardware danificado, algo com renderização, o mouse não poderia ter o DPI muito alto.

O Winamp era um processo único, que causava conflitos com outros processos e dava erro de renderização no Windows, exibia vários quadradinhos, várias janelas do Winamp, vários erros na tela.

Citado o exemplo do edge que é um navegar baseado em chrome, onde cada aba é uma thread dentro do processo do edge.

Uma thread é um processo mais leve, muito mais fácil de ser criado e de ser terminado, porém tem acesso a recursos de IO de Hardware, como acesso a discos, webcam, e também outros recursos do sistema operacional, até mesmo executar outros componentes do sistema operacional.

As threads do edge ou de outros serviços, nascem e morrem durante o processo de vida de um serviço.

Os sistemas operacionais Windows de hoje em dia, conseguem lidar de uma maneira melhor com as threads e processos.

Devido sua complexidade há uma necessidade muito grande de se estudar threads, as threads por exemplo possuem em seu subprocesso que é super rápido chamado de fibra, que é a menor unidade única e exclusiva de execução. No Java ainda está sendo implementado esse subprocesso das threads.

Essa fibra escola a thread que escalona o processo.

Existe duas maneiras de se criar threads, uma delas é utilizando herança e a outra é implementando a classe runable.

O run() é o método responsável por executar as threads é deve ser usado com @Override.

Observar a execução de monothreads com multithreads.

Observar as normas de utilização do Java pois a diferença de uma classe dentro da outra, para se construir threads, principalmente utilizando herança, o que se deve evitar, sendo a melhor maneira de usar threads a implementação ao invés da herança.

Implements Runnable. Por possuir o método run() , sendo a maneira correta de expressar a menor unidade de código, pois a classe Thread possui o método run() por implementar Runnable.

Observar que no Java você deve depender de uma interface e não de uma herança.

Em programação funcional usar a maneira funcional de se construir um contador ao invés do for clássico:

```
IntStream.rangeClosed ( 1, 5 ).forEach ( numero → {
        Final var minhaThread = new Thread ( new Contador( ) )
        minhaThread.start( );
)}
```

CONCORRÊNCIA

As threads executam em paralelo concorrendo entre si, porém é possível enfileirar threads e executar uma após a outra, caso seja desejado,

```
Public record Contador( int thread ) implements Runnable{
```

```
    @Override
    Public void run(){


    }
}
```

Concorrência é um conceito onde várias threads tentam acessar o mesmo recurso.

Há uma problemática nisso, como por exemplo causar deadlock de threads travadas, onde o processo não consegue finalizar as threads.

O Concorrência aponta diversos tipos de problemas, como o de sincronização, que pode não gerar o valor esperado, por estarem concorrendo pelo mesmo recurso e não existir sincronismos entre objetos.

O volatile resolve esse problema, ele tem o recurso que impede que o conteúdo de uma threads seja copiado pro cache, porém isso torna as coisas mais lentas, e ainda não resolve o problema da escrita de um recurso concorrido por várias threads.

O synchronized resolve esse tipo de problema, pois sincroniza o valor atual com as demais threads concorrentes.

Em todo decorrer da aula foi mostrado a evolução de um código, demonstrando solução de problemas de concorrência, concorrência de leitura, e concorrência de escrita.

Sincronismo de verdade não usa Join.

-AtomicInteger → é um valor atômico que basicamente vai se auto gerenciar para que a concorrência não afete a contabilização, o que também resolve o problema de sincronismo.