



C125 – Programação Orientada a Objetos com Java

Threads

Prof. Phyllipe Lima
phyllipe@inatel.br

1



Agenda



 Threads

 Exercícios

2

Executando em Paralelo



- ☞ Em várias situações, precisamos “executar duas coisas ao mesmo tempo”
Imagine um jogo bem grande, estilo GTA V (**Grand Theft Auto**). É necessário carregar texturas e modelos 3D para formar o mapa. Esse processo pode ser demorado (para os padrões atuais). Se o usuário ficar alguns segundos sem resposta é bastante desconfortável.
- ☞ Assim, é interessante que uma barra de progresso seja mostrada enquanto o jogo é carregado. Isto é, ao mesmo tempo!

Executando em Paralelo



- ☞ No geral, sempre fazemos várias coisas, **simultaneamente**, enquanto usamos o computador: Usar o navegador, ouvir música e jogar campo minado!
- ☞ Quando falamos em vários programas distintos, o próprio sistema operacional gerencia através de processos paralelos e usa com eficiência as várias CPUs existentes. Isso é tarefa do escalonador.
- ☞ Mas e um único programa? Como podemos quebrar em tarefas paralelas?
- ☞ Estamos falando de **Threads**

Threads no Java



O Java nos oferece a classe ***java.lang.Thread*** para criarmos linhas de execução paralela. A classe ***Thread*** recebe como argumento um objeto (instância) com o código que desejamos rodar. Considere uma classe hipotética para gerar o mapa do GTA V e outra para a barra de progresso (tela de carregamento)

```
public class GerarMapa {
    public void rodar() {
        //Logica para gerar o mapa
    }
}
```

```
public class GerarBarraProgresso {
    public void rodar() {
        //Logica para gerar a barra de progresso e
        //atualizar constantemente
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

5

5

Agora devemos ir no método ***main*** para iniciarmos a execução em paralelo



```
public static void main(String[] args) {
    //Thread para o mapa
    GerarMapa geraMapa = new GerarMapa();
    //Nao compila
    //Como que a thread saberá chamar o método rodar?
    Thread threadMapa = new Thread(geraMapa);
    threadMapa.start();

    //Thread para a barra de progresso
    GerarBarraProgresso geraBarra = new GerarBarraProgresso();
    //Nao compila
    //Como que a thread saberá chamar o método rodar?
    Thread threadBarra = new Thread(geraBarra);
    threadBarra.start();
}
```

- Criamos uma instância de Thread para cada trecho que queremos executar em paralelo, isto é, Threads diferentes!
- No exemplo, criamos uma para o mapa e outra para a barra de progresso.
- Para cada Thread, passamos como parâmetro para o construtor, a instância do objeto que contém a tarefa que desejamos executar em paralelo.
- Mas o código não compila! Como a Thread sabe qual método executar?

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

6

6

A Interface Runnable



- ☕ Não podemos passar qualquer instância para o construtor de Thread. Essa instância precisa **obedecer um contrato**. Isso nos lembra uma **interface**. Toda classe que **implementa** uma **interface** precisa obrigatoriamente implementar seus métodos. Isso é como se estivesse assinando um contrato.
- ☕ A interface **Runnable** tem o método **run()**, que deve ser implementado nas classes. Isso caracteriza o contrato.
- ☕ A classe **Thread** recebe como parâmetro no construtor, instâncias de classes que **implementam** a interface **Runnable**

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

7

7

A Interface Runnable



- ☕ Primeiro fazemos as nossas classes implementarem a interface **Runnable** (classes que desejamos executar paralelamente)

```
public class GerarMapa implements Runnable{
    //Usamos o metodo run para isso
    @Override
    public void run() {
        //Logica para gerar o mapa
    }
}
```

```
public class GerarBarraProgresso implements Runnable {
    @Override
    public void run() {
        //Logica para gerar a barra de progresso e
        //atualizar constantemente
    }
}
```

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

8

8



```
public static void main(String[] args) {

    //Thread para o mapa
    GerarMapa geraMapa = new GerarMapa();

    //Compila! Pois agora a classe GerarMapa
    //implementa a interface Runnable
    Thread threadMapa = new Thread(geraMapa);
    threadMapa.start();

    //Thread para a barra de progresso
    GerarBarraProgresso geraBarra = new GerarBarraProgresso();
    //Compila! Pois agora a classe GerarMapa
    //implementa a interface Runnable
    Thread threadBarra = new Thread(geraBarra);
    threadBarra.start();
}
```

Como as classes GerarMapa e GerarBarraProgresso implementam a interface **Runnable**, podemos passar suas instâncias para o construtor da classe Thread!



Herdando a Classe Thread

Outra forma de utilizar Threads é herdarmos da classe **Thread**. Uma vez que essa classe também implementa **Runnable** podemos sobrescrever o método **run()**.

```
public class GerarMapa extends Thread{

    @Override
    public void run() {
        //Logica para gerar o mapa
    }

}
```



Herdando a Classe Thread

☞ Agora podemos executar diretamente o próprio método `run()`.

```
public static void main(String[] args) {
    GerarMapa gerarMapa = new GerarMapa();
    gerarMapa.start();
}
```

☞ Apesar de ser um código mais simples, você está usando herança apenas por "preguiça" (herdamos um monte de métodos mas usamos apenas o `run`), e não por polimorfismo, que seria a grande vantagem.

☞ Prefira implementar **Runnable** a herdar de **Thread**.



Ordem das Threads

☞ Quando temos várias **threads** sendo executada, como sabemos a sua ordem de execução?

☞ Não sabemos! Quem irá decidir quando as threads serão executadas é o sistema operacional.

☞ Imagine uma CPU com dois núcleos. Isso significa que temos no máximo dois programa executando paralelamente. Mas quando usamos o computador temos vários programas abertos.

Ordem das Threads



- ☞ O **escalonador**, sabendo que apenas uma coisa pode ser executada de cada vez, pega todas as threads que precisam ser executadas e faz o processador ficar alternando a execução de cada uma delas. A ideia é executar um pouco de cada thread e fazer essa troca tão rapidamente que a impressão que fica é que as coisas estão sendo feitas ao mesmo tempo.
- ☞ Nós não controlamos essas escolhas (embora possamos dar "dicas" ao escalonador). Por isso que nunca sabemos ao certo a ordem em que programas paralelos são executados.

Exercício




- ☞ Crie um programa com três threads, cada uma imprimindo 1000 números inteiros.
- ☞ Observe a ordem que os números estão sendo impressos. Coloque um identificador para mostrar qual thread está sendo executada.

Vídeo Aula




 <https://youtu.be/zbLTzpGF56I>

 OBS: 2020/1


Material Complementar



 Capítulo 18 da apostila FJ-11

 Até o item 18.3



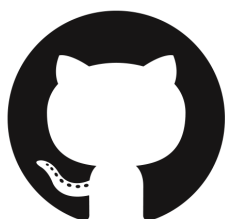
 Vídeo Aula Prof. Eduardo Guerra

 <https://youtu.be/gtnn-3fq5Sw>

Resolução dos Exercícios



<https://github.com/phillima-inatel/C125>



C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

17

17

Inatel

C125 – Programação Orientada a Objetos com
Java

Threads

Prof. Phyllipe Lima
phyllipe@inatel.br



18