Laboratório 12
Introdução ao uso de threads em Java

Computação Concorrente (MAB-117)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ — 14 de janeiro de 2016

Introdução

O objetivo deste Laboratório é aprender como criar programas concorrentes em Java. Para cada atividade, siga o roteiro proposto e responda às questões colocadas.

Atividade 1

Objetivo: Mostrar como criar um programa concorrente em Java. Em Java, a classe `java.lang.Thread` oferece métodos para criar threads, iniciá-las, suspendê-las e esperar pelo seu término.

O primeiro passo para criar uma aplicação concorrente em Java é **criar uma classe que implementa a interface Runnable**. Essa interface define o método **run()**, responsável pelo código que deverá ser executado pela thread.

O segundo passo é **transformar o objeto** `Runnable` **em uma thread**, chamando o construtor da classe java.lang.Thread com o objeto `Runnable` como argumento.

O terceiro passo é iniciar as threads criadas, usando o método **start()** da classe Thread.

Abra o arquivo **HelloThread.java** e siga o roteiro abaixo.

1. Leia o programa e tente entender o que ele faz (acompanhe a explanação da professora).
2. Compile o programa fazendo `javac HelloThread.java` no terminal.
3. Execute o programa **várias vezes** (fazendo `java HelloThread`) e observe os resultados impressos na tela. **Há mudanças na ordem de execução das threads? Por que isso ocorre?**
4. Descomente as linhas 43-45 e compile o programa novamente.
5. Execute o programa **várias vezes** e observe os resultados impressos na tela. **Qual alteração na execução da aplicação pode ser observada e por que ela ocorre?**

Atividade 2

Objetivo: Mostrar outra forma de criar threads em Java.

Roteiro: Outra forma de criar programas concorrentes em Java é estendendo a classe Thread. Abra o arquivo **OlaThread.java** e siga o roteiro abaixo.

1. Primeiro, encontre as principais diferenças em relação ao programa `HelloThread.java`. Acompanhe a explanação da professora.
2. Compile e execute o programa **várias vezes**, e observe os resultados impressos.

Atividade 3

Objetivo: Mostrar um exemplo de aplicação com threads e memória compartilhada em Java.

Roteiro: Abra o arquivo **TIncrementoBase.java** e siga o roteiro abaixo.

1. Leia o programa para entender o que ele faz. Acompanhe a explanação da professora. **Qual é a seção crítica do código?** Qual saída é esperada para o programa (valor final de **s**)?
2. Compile o programa, execute-o **várias vezes** e observe os resultados impressos na tela. **Os valores impressos foram sempre o valor esperado? Por que?**

Atividade 4

Objetivo: Mostrar como implementar **exclusão mútua** em Java.

Roteiro: Ainda no arquivo **TIncrementoBase.java**:

1. Comente as linhas 17-19; e descomente as linhas 21-25.
2. Acompanhe a explanação da professora sobre o uso de `synchronized` em Java.
3. Compile o programa, execute-o **várias vezes** e observe os resultados impressos na tela. **Os valores impressos foram sempre o valor esperado? Por que?**

Atividade 5

Objetivo: Implementar um programa concorrente para somar dois vetores, fazendo C = A + B. Considere que o número de threads é **menor ou igual** ao número de elementos dos vetores. Divida a tarefa entre as threads de forma balanceada.

Roteiro:

1. Projete uma classe em Java para conter um vetor e os métodos de acesso a ele (construtor, inicialização, impressão, tamanho do vetor, alteração de um elemento).
2. Qual(is) argumento(s) deverá(ão) ser passado(s) para cada thread manipular uma parte do vetor?
3. Na thread *main* crie uma instância da classe vetor (para cada vetor A, B e C), leia os valores iniciais dos vetores A e B da entrada padrão, crie e dispare as threads, aguarde todas as threads terminarem e imprima os valores finais do vetor C.
4. Teste seu programa.

Empacote e envie os programas para correção Crie um diretório e o nomeie juntando seu "primeiro" e "último" nome. Copie pra dentro desse diretório o código fonte da atividade **5**. Comprima o diretório (ex., `zip -r JoseSilva.zip JoseSilva/`) e envie o arquivo comprimido para o endereço de email **computacao.concorrente.ufrj@gmail.com** com subject "**CompConc Lab12**".

O email deve ser enviado até amanhã, dia 15/01