Ola, meus amigos!

Tenho uma dúvida que está me dando muito trabalho, já fiz várias pesquisa na internet, vídeo aulas e afins, mas a dúvida persisti ou ao menos parte dela.

Então vou lhes perguntar.

Como eu posso sincronizar um Objeto entre várias Thread’s???

Ok, para contextualizar, eu sei como criar uma Thread (estendendo Thread ou implementando Runnable), sei como sincronizar um Objeto (dados) pertencente a uma thread, pode ser sincronizado:

1. Criando várias threads a partir do MESMO objeto da classe que implementa Runnable e declarando os métodos que alteram os dados como synchronized
2. Criando várias threads a partir de VÁRIOS objetos da classe que implementa Runnable e declarara os métodos que alteram os dados como synchronized, e os dados como static

Mas o que eu quero é alterar os dados que estão em uma classe diferente da classe que implementa Runnable, por exemplo e também não são static  
  
Suponha que eu tenho um Objeto servidor de uma classe SERVIDOR

E vários objetos escravos de uma classe ESCRAVO

Então o servidor inicia seu sua atividade abrindo um socket para escutar os escravos...

Daí quando um escravo se conecta o servidor cria uma Thread que vai gerir a conexão com o escravo, então o escravo envia uma mensagem contendo uma lista de arquivos e então essa mensagem é captura e processada pela Thread que o servidor criou para gerir a conexão...

Como eu poderia então fazer com que essa Thread colocasse os dados em uma lista de arquivos do servidor um atributo do objeto servidor, mas de forma que somente um método desse pudesse está ativo por vez, ou seja, embora existam varias threads e elas trabalhem separadamente na hora de “gravar” uma informação no servidor elas deverias sincronizar...  
Eu conseguir esse efeito mas acho que ainda não está da melhor forma, o que eu fiz foi fazer cada Thread atualizar sua própria lista de arquivos e então chamar uma método do servidor que é sincronizado para atualizar a lista do servidor, MAS, achei essa solução meia boca, eu pensei em usar o wait e notify mas, fico a pensar se não há uma forma apenas usando synchronized... Então denovo!

O que eu quero é que várias threads alterem dados de uma outra classe mas que essa alteração seja sincronizada, (uma após as outra), por exemplo se eu tenho o método adicionarParaServidor, então uma Thread só pode iniciar este serviço depois uma outra tiver terminado... eu não estou conseguindo fazer isso!

Para ficar ainda mais claro segue o exemplo de código abaixo:

Essa classe deveria ter o método processarLista “Sincronizado”

public class Escravo implements Runnable {

private static int base = 0;

@Override

public void run() {

processarLista();

}

public synchronized void processarLista() {

for(int i = 0; i < 5; i++) {

try {

System.out.println(Thread.currentThread().getName() + " aguardando processamento...");

Thread.sleep(1000);

}

catch (InterruptedException ex) {

System.err.println("Erro ao executar sleep.");

}

}

}

}

Está e a classe Main

public class Main {

public static void main(String[] args) throws InterruptedException {

System.out.println("Estudo de Threads. Executando...");

new Thread(new Escravo(), "Escravo 1").start();

new Thread(new Escravo(), "Escravo 2").start();

new Thread(new Escravo(), "Escravo 3").start();

new Thread(new Escravo(), "Escravo 4").start();

}

}

Isso resultada na saída...

Escravo 1 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 4 aguardando processamento...

O que eu não entendo é se o método processarLista lista é sincronizado por que não se repeti todo o for da primeira thread, depois o da segundo, terceira e assim por diante, por que elas continuam a correr paralelamente...

Contudo se eu fizer:

public class Main {

public static void main(String[] args) throws InterruptedException {

System.out.println("Estudo de Threads. Executando...");

Escravo escravo = new Escravo();

new Thread(escravo, "Escravo 1").start();

new Thread(escravo, "Escravo 2").start();

new Thread(escravo, "Escravo 3").start();

new Thread(escravo, "Escravo 4").start();

}

}

Aí então a saída é:  
  
Escravo 1 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 1 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 4 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 2 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 3 aguardando processamento...

Escravo 3 aguardando processamento...

Exatamente o que eu queria, a execução não está em “1, 2, 3, 4” MAS cada for e executado todo e só depois é passado para outro Thread

Mas isso só ocorre por que eu estou criando as várias Threads a partir do mesmo objeto da classe que implementa Runnable... O que eu quero saber é como conseguir o mesmo feito (“a forma correta”) mas a partir de objetos diferentes...   
  
Eu acredito que minha compreensão sobre como as Threads funcionam será muito ajudada se eu puder entender isto.

Eu sei que vão dizer, mas tu quer que as Thread’s trabalhem sequencialmente... ???

SIM, isso é importante para o meu entendimento de como essa execução funciona e de como as Threads funcionam, depois vou estender para algo semelhante ao m pool de tarefas...

Grato e desculpe o tamanho do tópido.

public class ContaConjunta {

private int saldo = 100;

public int getSaldo() {

return saldo;

}

public synchronized void sacar(int valor, String cliente) {

if(saldo >= valor) {

int saldoOriginal = saldo;

System.out.println(cliente + " vai sacar.");

try {

System.out.println(cliente + " aguardando processamento...");

Thread.sleep(1000);

}

catch (Exception ex) {

System.err.println("Erro no método sleep.");

}

saldo -= valor;

String msg = String.format("%s SACOU %d, [Saldo Original = %d -> Saldo Atual = %d]", cliente, valor,

saldoOriginal, saldo);

System.out.println(msg);

}

else {

System.out.println("Saldo insuficiente para " + cliente);

}

}

}

E Supondo que eu use a seguinte classe que implementa Runnable para criar várias Thread’s que irão operar em um objeto da classe ContaConjunta.

public class Compras implements Runnable {

private static ContaConjunta conta = new ContaConjunta();

@Override

public void run() {

for(int i = 0; i < 5; i++) {

conta.sacar(20, Thread.currentThread().getName());

if(conta.getSaldo() < 0) {

System.out.println("Estourou para " + Thread.currentThread().getName());

}

}

}

}

Então eu faço uma Main que crie as Várias Thread’s

public class Main {

public static void main(String[] args) {

System.out.println("Estudo de Thread's");

Compras compras = new Compras();

new Thread(compras, "Pai").start();

new Thread(compras, "Mãe").start();

new Thread(compras, "Filho").start();

}

}

Neste caso haverá uma sincronização uma vez que o método sacar em ContaConjunta é synchronized e TODAS as Thread’s são criadas a partir do mesmo objeto (instância da classe que implementa Runnable -> Compras)  
  
Até ai tudo bem, MAS caso que queira que cada Thread seja criada a partir de um objeto distinto da classe que implementa Runnable, como em:

public class Main {

public static void main(String[] args) {

System.out.println("Estudo de Thread's");

new Thread(new Compras(), "Pai").start();

new Thread(new Compras(), "Mãe").start();

new Thread(new Compras(), "Filho").start();

}

}

Bem neste caso eu estou criando 3 objetos de Compras, e como cada objeto tem um ContaConjunta existirão 3 contas conjuntas e elas estão “isoladas” umas das outras, para que eu possa ter o mesmo efeito de sincronização devo então declarar o objeto conta corrente como static dentro da classe Compras, fazendo desta forma que haja apenas um objeto deste tipo e logo todas a Thread’s acessarão este objeto e tudo ficará sincronizado também.