Sistemas Operativos
2021 / 2022

Licenciatura em Engenharia Informática

Lab. 08 – Criação de threads em Java

Nesta aula pretende-se que os alunos fiquem com uma noção prática da criação de threads utilizando a linguagem de programação Java.

Ex. 1 – Criação de threads

O seguinte programa exemplifica a criação e utilização de *threads* em Java.

```
1  public class SimpleThread extends Thread {
2
3      @Override
4      public void run() {
5          for (int i=0; i<5; i++) {
6              System.out.println(this.getName() + ": " + i);
7          }
8      }
9
10     public static void main(String[] args) {
11         SimpleThread thread0 = new SimpleThread();
12         SimpleThread thread1 = new SimpleThread();
13
14         thread0.start();
15         thread1.start();
16     }
17 }
```

Crie um novo projecto Java no seu IDE e coloque o código anterior num ficheiro de nome *SimpleThread.java*. O *output* deverá ser semelhante ao seguinte:

```
Thread-0: 0
Thread-0: 1
Thread-1: 0
Thread-1: 1
Thread-1: 2
Thread-0: 2
Thread-1: 3
Thread-0: 3
Thread-1: 4
```

```
Thread-0: 4
```

De uma forma geral, o programa cria dois objectos do tipo *SimpleThread* e depois invoca o método *start()* de modo a iniciar a execução das threads. O código que é executado paralelamente será o código implementado no método *run()*.

a) Execute o programa várias vezes e verifique se a ordem de execução das threads é previsível ou se se altera de execução para execução.

b) Recorrendo ao método *sleep(int millis)* altere o programa de modo a que as threads suspendam, por 500 ms, a sua execução entre cada iteração do ciclo *for*. Comente as diferenças em relação ao resultado anterior.

c) Usando o código original (i.e., sem *sleep*), utilize o método *setPriority(int)* de modo a que a execução da segunda thread (thread-1) tenha prioridade sobre a primeira.

d) Modifique o programa de modo a que a segunda thread só inicie a sua execução após o fim da primeira thread. Utilize o método *join()* para esperar o fim da primeira thread.

Ex. 2 – Criação de múltiplas threads

Pretende-se implementar um programa em Java que permita criar várias threads. Para cada thread, esse programa deverá escrever o nome da thread, aguardar um determinado tempo aleatório (entre 0 e 1 segundos) e depois deverá escrever a informação que vai sair. O *output* para cinco threads deverá ser semelhante ao seguinte:

```
Thread-0
Thread-1
Thread-2
Thread-3
Thread-4
Thread-2 is exiting..
Thread-1 is exiting..
Thread-4 is exiting..
Thread-0 is exiting..
Thread-3 is exiting..
```

a) Implemente o programa acima mencionado para uma thread apenas.

b) Modifique o programa anterior de modo a permitir criar várias threads. Utilize um ciclo *for* para ir criando e inicializando as várias threads.

c) Coloque uma mensagem “Program is ending” após a inicialização das várias threads e verifique que a mensagem aparece antes do fim das várias threads.

d) Modifique o programa de modo a que a mensagem de fim do programa apareça apenas após o fim das várias threads. Considere guardar os objectos das *threads* num array e esperar, num ciclo *for* posterior, que todas as threads terminem, usando o método *join()*.

Ex. 3 – Utilização de propriedades em threads

Considere a seguinte classe que permite somar dois números recorrendo à utilização de threads. De uma forma simples, visto que o método *run()* apenas pode aceder às propriedades do próprio objecto, torna-se necessário passar os dados por parâmetro no construtor, executar a thread e esperar que termine de modo a obter o resultado da soma.

```
1  public class SumThread extends Thread {
2
3      int num1, num2, res;
4
5      public SumThread(int num1, int num2) {
6          this.num1 = num1;
7          this.num2 = num2;
8      }
9
10     @Override
11     public void run() {
12         res = num1 + num2;
13     }
14
15     public static int sum(int num1, int num2) throws InterruptedException {
16         SumThread thread = new SumThread(num1, num2);
17         thread.start();
18         thread.join();
19         return thread.res;
20     }
21
22     public static void main(String[] args) throws InterruptedException {
23         System.out.println(sum(10, 20));
24     }
25 }
```

a) Implemente o programa anterior e verifique o seu correcto funcionamento.

b) Modifique o programa anterior de modo a que a função *sum(int array[])* some um array de números inteiros invés de dois números apenas.

c) Modifique o programa anterior de modo a que a função *sum()* utilize duas threads, onde uma das threads soma a primeira metade do array e a outra thread soma a segunda metade do array. Por fim, a função *sum()* deverá retornar a soma de ambas as metades. Considere passar o array e os índices inicial e final de cada thread como parâmetros para o construtor.

d) Generalize o programa anterior de modo a que o número de threads a utilizar seja passado como parâmetro no método *sum(int array[], int nthreads)*. Utilize um ciclo *for* para criar e inicializar as várias threads, e outro ciclo para esperar pelo fim das várias threads e ir somando os resultados.

e) Crie o método *static max(int array[], int nthreads)* para encontrar o maior número no array usando várias threads. Considere usar uma nova variável que vá guardando o valor máximo à medida que percorre o array.

(fim de enunciado)