

Sistemas Distribuídos - Aulas Práticas -

Universidade do Minho
2019/2020

Material Aulas

- Guiões com exercícios
- Java (SDK 7+)
 - OpenJDK, Oracle Java
- IDE:
 - **IntelliJ IDEA**, Eclipse, Netbeans

Aquecimento

- Programa Java que substitua o comando 'echo' :
- Ler linha a linha da consola e imprimir para stdout
- InputStreamReader, BufferedReader

Threads

- Fios de execução concorrentes de um programa
- Um processo tem uma ou mais threads
- Partilham recursos e comunicam entre si através de memória partilhada
- Analogia:
 - As threads de um processo são como vários cozinheiros que seguem as instruções do mesmo livro de culinária, mas não necessariamente todos na mesma página.

Threads em Java

- `java.lang.Runnable`
 - interface implementada por classes cujas instâncias representam threads
 - classes que implementem esta interface têm que implementar o método `run()`
- `java.lang.Thread`
 - implementa `java.lang.Runnable`
 - classes que estendam `Thread` devem re-implementar o método `run()`
 - outros métodos relevantes: `start()`, `sleep()`, `join()`

Exemplo HelloWorld:

```
public class HelloRunnable implements Runnable {  
    public void run() {  
        System.out.println("Hello from a thread!");  
    }  
  
    public static void main(String args[]) {  
        (new Thread(new HelloRunnable())).start();  
    }  
}
```

Exemplo HelloWorld 2:

```
public class HelloRunnable2 implements Runnable {
    int n;

    public void run() {
        System.out.println(n);
    }

    HelloRunnable2(int a) {
        n=a;
    }

    public static void main(String args[]) {
        HelloRunnable2 r222 = new HelloRunnable2(222);
        HelloRunnable2 r111 = new HelloRunnable2(111);
        Thread t1=new Thread(r222);
        Thread t2=new Thread(r111);
        System.out.println("Antes");
        t1.start();
        t2.start();
        System.out.println("Depois");
        try {
            t2.join();
            t1.join();
        } catch (InterruptedException e) {}
        System.out.println("Fim");
    }
}
```

Exemplo HelloWorld 3:

```
public class HelloRunnable3 implements Runnable {
    int n;
    public void run() {
        System.out.println(n);
        this.set(111);
    }
    HelloRunnable3(int a) {
        n=a;
    }

    public void set(int b) { n=b; }

    public static void main(String args[]) {
        HelloRunnable3 r=new HelloRunnable3(222);
        Thread t1=new Thread(r);
        Thread t2=new Thread(r);
        System.out.println("Antes");
        t1.start();
        t2.start();
        System.out.println("Depois");
        try {
            t2.join();
            t1.join();
        } catch (InterruptedException e) {}
    }
}
```


● Exemplo HelloWorld 3:

T₁

print(n)

n=111

T₂

print(n)

n=111

Output:

222

111

● Exemplo HelloWorld 3:

T₁

print(n)

n=111

T₂

print(n)

n=111

Output:

222

222

Exercícios

- 1) Escreva um programa que crie N threads, em cada uma escreva os números de 1 a I, e depois espere que estas terminem.

Nota: **Usar um array para guardar referências das N threads e poder invocar:**

- **new Thread()**
- **start()**
- **join()**

Exercícios

- 2) Modifique o programa para as N threads terem acesso a um único objecto de uma classe Counter. Cada thread deverá agora incrementar I vezes o contador. Escreva duas versões: uma em que cada thread invoca um método increment da classe Counter e outra em que as threads acedem directamente a uma variável de instância.
- 3) Fazendo a thread principal escrever o valor do contador depois de as outras threads terem terminado, corra várias vezes o programa, para diferentes valores de N e I e observe o resultado produzido, em ambas as versões.


Exercícios 2 e 3

Counter

```
public int count  
Counter()  
increment()  
getCounter()
```

Incrementor

implements
Runnable



```
private Counter c  
Incrementor(c, l)  
run()
```

Main

```
int N, int l  
threadArray[N]  
  
main(...)  
//cria e inicia threads  
//imprime contador
```