

Programação Concorrente

Java, Erlang e Python

Grupo

Erica Amoedo

Matheus Soares

Vinícius Pessoa

Java

Java

- Utiliza a classe Thread para gerenciar o multiprocessamento;
- Uma classe deve herdar a classe Thread e sobrescrever o método run();
- Para que uma thread seja iniciada, chama-se o método start();
- O método start() executa o método run();
- O método start() garante que uma thread diferente será criada;

```
import java.util.Random;

public class Corredor extends Thread{
    private String nome;
    private int tempo;
    private Random gerador;

    public Corredor(String nome){
        this.nome = nome;
        gerador = new Random();
    }

    public String getNome(){
        return nome;
    }

    public int getTempo(){
        return tempo;
    }

    public void calcularVel(){
        this.tempo = gerador.nextInt(1001);
    }
}
```

```
@Override
public void run(){
    String saida = nome+": ";

    calcularVel();

    for (int j = 0; j < (tempo/100); j++)
    ){
        saida += "-";
    }

    saida += "|" + tempo;

    // Espera o tempo em ms
    try {
        Thread.sleep(tempo);
    }
    catch (InterruptedException ex) {
        Thread.currentThread().interrupt
    }

    System.out.println(saida);
    saida = "";
}
}
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class main {

    public static void main(String[] args) {
        int n_Tiros = 10;
        int n_Corredores = 5;
        int menorTempo = 10000;
        int[] tempos = new int[n_Corredores];
        Corredor[] corredores = new Corredor[n_Corredores];

        for (int i = 0; i < n_Tiros; i++){
            // Instancia corredores
            Corredor joao = new Corredor("João ");
            Corredor maria = new Corredor("Maria ");
            Corredor felipe = new Corredor("Felipe ");
            Corredor carlos = new Corredor("Carlos ");
            Corredor daniela = new Corredor("Daniela");

            // Adiciona corredores no array
            corredores[0] = joao;
            corredores[1] = maria;
            corredores[2] = felipe;
            corredores[3] = carlos;
            corredores[4] = daniela;
```

```
        System.out.println("\n\nCorrida " + (i+1) + "\n");

        // Inicio da corrida
        joao.start();
        maria.start();
        felipe.start();
        daniela.start();
        carlos.start();

        // Espera de 2s para que a corrida acabe e os resultados parciais sejam mostrados
        try {
            Thread.sleep(2000);
        } catch (InterruptedException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }

        // Soma as tempos
        tempos[0] += joao.getTempo();
        tempos[1] += maria.getTempo();
        tempos[2] += felipe.getTempo();
        tempos[3] += carlos.getTempo();
        tempos[4] += daniela.getTempo();

    }
}
```

```
// Imprime o total de tempo em que cada um levou
System.out.println("\n\nTOTAL DE TEMPOS (ms)\n");
for (int i = 0; i < n_Corredores; i++){
    System.out.println(corredores[i].getNome() + ": " + tempos[i]);
    if (tempos[i] < menorTempo){
        menorTempo = tempos[i];
    }
}
System.out.println("\nMenor Tempo = " + menorTempo);

for (int i = 0; i < n_Corredores; i++){
    if (tempos[i] == menorTempo){
        System.out.println("\nVencedor(a): "+corredores[i].getNome());
    }
}
}
```


Erlang

Erlang

- Utiliza *processos* que não compartilham dados;
- Processos são iniciados com o método *spawn(módulo, função, [argumentos])*;
- O método *spawn* retorna um identificador de processo (PID);
- Processos se comunicam através de mensagens;
- Mensagens são enviados com o operador ! - *PID ! mensagem*;
- Mensagens são recebidas utilizando o bloco *receive... ...end*;

```
runner(Name, PID_lap) -> runner(Name, 1, 0, PID_lap).

runner(Name, 11, Time, _) -> io:format("~s finished with ~p seconds!~n", [Name, Time
]);
runner(Name, Lap, Time, PID_lap) ->
  Num = rand:uniform(7),
  timer:sleep(Num*1000),
  PID_lap ! {self(), Name, Lap, Num},

  receive
    go -> runner(Name, Lap + 1, Time + Num, PID_lap)
  end.
```

Processso *runner*

```

lap(First) ->
  if
    First ->
      PID_lap = self(),
      spawn(main, runner, ["João", PID_lap]),
      spawn(main, runner, ["Maria", PID_lap]),
      spawn(main, runner, ["Felipe", PID_lap]),
      spawn(main, runner, ["Carlos", PID_lap]),
      spawn(main, runner, ["Daniela", PID_lap]);
    true -> ok
  end,

  [
    receive {PID1, Name1, Lap1, Num1} -> io:format("~s finished the lap ~p with ~p seconds~n", [Name1, Lap1, Num1]) end,
    receive {PID2, Name2, Lap2, Num2} -> io:format("~s finished the lap ~p with ~p seconds~n", [Name2, Lap2, Num2]) end,
    receive {PID3, Name3, Lap3, Num3} -> io:format("~s finished the lap ~p with ~p seconds~n", [Name3, Lap3, Num3]) end,
    receive {PID4, Name4, Lap4, Num4} -> io:format("~s finished the lap ~p with ~p seconds~n", [Name4, Lap4, Num4]) end,
    receive {PID5, Name5, Lap5, Num5} -> io:format("~s finished the lap ~p with ~p seconds~n", [Name5, Lap5, Num5]) end
  ],
  io:format("~n"),

  PID1 ! go, PID2 ! go, PID3 ! go, PID4 ! go, PID5 ! go,
  lap(false).

start() -> spawn(main, lap, [true]).


```

Processo *lap*

Python

Python

- É usada a biblioteca “threading”;
- Podem ser criadas pela herança de classes;
- Ou pelo uso da função;
- Implementação similar ao Java;



```

import threading
import time
import random
import sys

def animacao(tempo):
    ani = "|/-\\"
    for i in range(tempo):
        time.sleep(0.05)
        sys.stdout.write("\r" + ani[i % len(ani)])
        sys.stdout.flush()

class Corredor(threading.Thread):
    resultado = []
    def __init__(self, nome):
        threading.Thread.__init__(self)
        self.nome = nome
        self.resultado = []

    def run(self):
        random.seed(time.time())
        tempo = random.uniform(2, 5)
        time.sleep(tempo)
        animacao(int(tempo))
        self.resultado.append(tempo*1000.0)

    def retornarResultado(self):
        return sum(self.resultado)

threads = []

```

```

joao = Corredor("João")
maria = Corredor("Maria")
felipe = Corredor("Felipe")
carlos = Corredor("Carlos")
daniela = Corredor("Daniela")

threads.append(joao)
threads.append(maria)
threads.append(felipe)
threads.append(carlos)
threads.append(daniela)

for t in threads:
    t.start()
for t in threads:
    t.join()

print(" volta completa")

for i in range(0, 9):
    for t in threads:
        t.run()
    for t in threads:
        t.join()
    print(" volta completa")

menorTempo = []

for t in threads:
    print("Nome: ", t.nome)
    print("Resultado (em ms): ", t.retornarResultado())
    menorTempo.append(t.retornarResultado())

print("Menor tempo: ", min(menorTempo))

```