Matéria:

* Revisão da matéria lecionada

1. **– Concorrência entre *threads***

Implemente um programa que simule transações sobre uma conta bancária partilhada pela Alice e pelo Bruno. Ambos têm um emprego, auferindo cada um deles um ordenado de 1250 euros mensais. Durante o mês, Alice e Bruno efetuam um conjunto de dez despesas cada um de valor aleatório entre 150 e 350 euros. O programa compreende duas classes: *BankAccount* e *JobAliceBruno.* A classe *BankAccount* contém informação sobre o saldo da conta, que é incrementado ou decrementado recorrendo a dois métodos *putMoney* e *getMoney.* Escreva o programa de forma a garantir que Alice e Bruno (dois *threads*) não retiram da conta nenhum valor superior ao saldo. Para isso, deve simular este cenário durante seis meses.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

1. – Semáforos em Java



Listagem 1: Classe *SemaphoreTest*

Considerando a classe *SemaphoreTest* apresentada na Listagem 1. Qual será o resultado esperado se se substituir às chamadas as funções *fun()* pelas seguintes:



import java.util.concurrent.\*;  
  
public class Exe2 {  
 static Semaphore *s* = new Semaphore(0);  
  
 public void fun(final char c, final int r) throws Exception {  
 new Thread(new Runnable() {  
 public void run() {  
 try {  
 System.*out*.println("acquire-" + r);  
 *s*.acquire(r);  
 System.*out*.println(c + "-" + r);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }).start();  
 Thread.*sleep*(500);  
 }  
  
 public static void main(String[] args) throws Exception {  
 Exe2 f = new Exe2();  
  
 f.fun('C', 7);  
 f.fun('B', 2);  
 f.fun('A', 5);  
  
  
 while (*s*.hasQueuedThreads()) {  
 Thread.*sleep*(1000);  
 System.*out*.println("release-1 available-" + (*s*.availablePermits() + 1));  
 *s*.release(1);  
 }  
 }  
}

Com a substituição das funções fun() do programa original para as indicadas acima pode ver que o resultado será Executar o C 7 vezes depois executar o B 2 vezes e por fim executar o A 5 vezes.

1. **– Sinalização entre *threads* em Java**

Elabore duas classes: **Produtor** e **Consumidor.** Estas classes deverão trocar informação entre elas recorrendo a um buffer com 5 carateres de capacidade máxima. À classe **Produtor** compete a geração de carateres e a sua escrita no buffer. À classe **Consumido**r compete a leitura de caracteres do *buffer*. A classe Produtor deve poder escrever até que encha o *buffer*, altura em que deve ser suspensa até que o buffer volte a ficar vazio. A classe Consumidor só devem conseguir ler do *buffer* quando este estiver completamente cheio, adormecendo quando este estiver vazio. O acesso ao *buffer* deve ser executando sempre em exclusividade de acesso, independentemente de ser um acesso de escrita ou leitura.

Teste a sua solução com uma frase à sua escolha.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente