ISCTE – Instituto Superior de Ciências do Trabalho e da Empresa

Programação Concorrente e Distribuída
Exame Modelo #1

Licenciatura em Engenharia Informática
Licenciatura em Informática e Gestão de Empresas
Licenciatura em Engenharia de Telecomunicações e Informática

Parte Teórico-prática (vale 12 valores)

Duração: 1h30m
Sem Consulta

Nome:
Número:
Curso:

1.1. Esta prova tem duas partes, uma parte teórica e uma parte teórico-prática.
1.2. Apenas os alunos que não se encontram em avaliação periódica fazem este exame.
1.3. Haverá um intervalo após a realização da parte teórica, sendo o enunciado da parte teórico-prática então distribuído.
1.4. **Complete** a sua identificação em todas as folhas, com nome, número e curso.
1.5. Responda nos espaços indicados para esse efeito. Se não tiver espaço complete a sua resposta no verso da folha ou nas folhas brancas em anexo.
1.6. A prova é sem consulta. Em cima da sua mesa deve ter apenas uma esferográfica e um documento de identificação. Os telemóveis devem estar desligados.
1.7. Qualquer troca de informações com colegas implica a anulação imediata da prova e o procedimento disciplinar correspondente.
1.8. Não há resposta a dúvidas após os primeiros vinte minutos de prova. Se tiver dúvidas em relação a alguma pergunta considere os pressupostos que entender de forma a justificar a sua resposta. Indique os pressupostos que assumir.

Número:
Nome:

[2 valores]

1.

Crie uma aplicação que mostra uma janela com três botões. Inicialmente, os botões estão desligados, o que significa que o botão não pode ser clicado (para isso use o método `setEnabled(false)` da classe `JButton`). Deve ser criado uma thread para cada botão que faz uma contagem descendente partindo de um número aleatório entre 1 e 10, a que corresponde o número de segundos que o botão permanecerá desligado. Para cada segundo da contagem descendente, o thread deve alterar o nome mostrado no botão para o número de segundos que restam até o botão ser ligado. Uma vez finda a contagem descendente, o thread deve ligar o botão (`setEnaled(true)`) e alterar o nome do botão para "Hit me!".

O nome do botão pode ser alterado através do método `setText(String newLabel)`, da classe `JButton`.

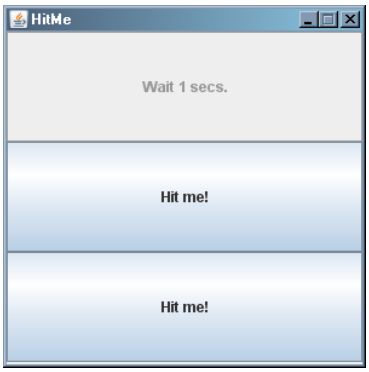
Em baixo ilustra-se um possível exemplo. Certifique-se que a sua aplicação mostra uma janela com uma disposição (layout) similar. O tamanho da janela é de 300 x 300.

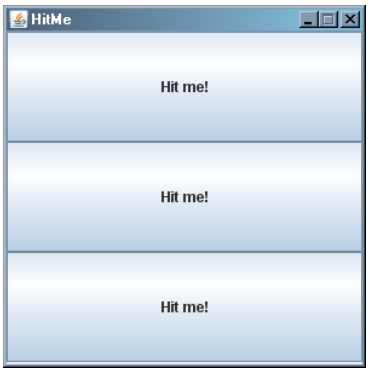
1 

2 

3 

4 

5 

6 

Número:
Nome:

[2 valores]
2. "Gerador de Pin's"

Num banco pretende-se construir uma aplicação que gera automaticamente os pins (personal identification number) dos novos cartões de multibanco a distribuir aos cliente. Para isso quer implementar uma aplicação com 4 threads que geram dígitos de uma forma aleatória. No entanto os dígitos devem ser introduzidos num recurso partilhado que deve criar o número. O primeiro dígito a ser inserido será o dígito com maior ordem de grandeza.

Escreva um programa para coordenar os quatro geradores de dígitos.

A aplicação deve gerar 1000 Pins de cada vez que for corrida a partira da consola (**não** tem interface gráfica, GUI).

[5 valores]
3. O DNA consiste numa sequência constituída por açúcar, grupos de fosfato e quatro bases A, T, C e G. Estas 4 bases constituem a informação genética do DNA. Realize um programa que conta a ocorrência de cada uma das letras A, T, C e G, respectivamente, em strings. Considere que tem na sua classe principal um método `getSequencedData()` que retorna uma `String` (assuma que este método já está implementado). Este método retorna novas strings sempre que for invocado, até um ponto em que não haverá mais strings e retornará `null`. Por questões de eficiência as strings devem ser processadas em paralelo, mas não deve haver em nenhum momento mais de 10 strings a serem processadas. Use um semáforo (veja sumário da API abaixo) para garantir que apenas existem 10 threads. Uma vez que todas as strings tenham sido processadas, os número totais de A, T, C e Gs, respectivamente, devem ser mostradas na consola (utilizando o `System.out.println(),` não utilize qualquer swing neste programa).

Recorde que o método `charAt(int pos)` retorna uma letra da string numa dada posição (por exemplo, numa dada string `s`, `s.charAt(0)` retorna a primeira letra e `s.charAt(5)` retorna a letra na posição 5, ou seja a sexta letra).

A seguinte classe deverá ser a sua classe principal. Implemente todas as funcionalidades necessárias com excepção do método `getSequencedData`.

Considere o seguinte exemplo da execução do programa para 3 cadeias de DNA:

"ATTTTAAAAGGGCGCGCGCCCCA"
"AAAC"
"CCCCCCCCCC"

Resultado:
```
A: 9 vezes
T: 4 vezes
C: 18 vezes
G: 6 vezes
```

Número:
Nome:

`Semaphore API:`

Constructor Summary

`Semaphore``(int permits, boolean fair)` Creates a `Semaphore` with the given number of permits and the given fairness setting.	

Method Summary

`void`	**`acquire`**`()` Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
`void`	**`release`**`()` Releases a permit, returning it to the semaphore.

```
class DNACounter {
          private String getSequencedData() { // do not implement}




}
```

[3 valores]

4. Considere que o contador de DNA da questão 3 deve receber strings de clientes numa rede.

Implemente na classe `DNACounter` o método `getSequencedData()` e ainda um um novo método `setupServer()`. O método `setupServer()` deve inicializar o servidor, ficando à espera da ligação de clientes. Cada vez que o método `getSequencedData()` é invocado a ligação com o cliente deve ser aceite e a string enviada pelo cliente deve ser retornada pelo método. Caso ocorra algum problema com a comunicação com o cliente, este método deve impossibilitar futuras ligações com clientes e retornar `null`.

Número:
Nome:

```
class DNACounter {
  // ponha aqui os novos atributos da classe




  private String setupServer() {
    // implemente este método como descrito




  }


  private String getSequencedData() {
    // implemente este método como descrito







  }
```