



Sincronização



5.1 Incremento e decremento concorrente

Programa uma classe contador que contenha os seguintes métodos:

- incrementar,
- decrementar e
- consulta do valor.

Programa também uma classe de processos ligeiros onde cada instância incrementa o valor do contador. O processo deve repetir esta ação 1000 vezes, após o que termina.

Teste o seu programa com quatro processos ligeiros e um contador.



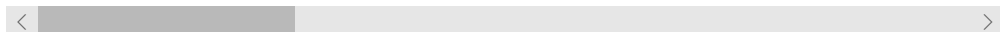
5.2 Pilha de inteiros

Implemente uma pilha de inteiros que permite um acesso concorrente.

A pilha tem os seguintes métodos:

boolean	empty() Verifica se a pilha está vazia
int	peek() Devolve o elemento no topo da pilha sem o remover
int	pop() Retira o elemento do topo da pilha
void	push(int) Adiciona um elemento ao topo da pilha
int	size() Devolve o número de elementos na pilha

Teste a implementação criando uma sub-classe de Thread que manipule a pilha. Lance 6 instâncias desta classe.



5.3 Conta bancária

Crie uma classe ContaBancaria e um método depositar.

```
class Account {  
    void deposit(int amount); // put money into the account  
  
    int getBalance(); // get the current balance of the  
    account  
}
```

Crie 10 Threads Cliente, cada um vai fazer depósitos contínuos de um valor aleatório (por exemplo entre 0 e 100€) até ser interrompido. Cada cliente guarda o total que ele próprio depositou.



O main, fará o seguinte:

1. Lança os 10 clientes
2. dorme 10 segundos,
3. interrompe todos os clientes,
4. espera que terminem (usando o join()),
5. imprime o saldo da conta
6. soma o total depositado por todos os clientes para confirmação.

Experimente com e sem sincronização do metodo deposit().



5.4 Slot Machine

Arquivos anexados:  [im1.jpg](#) (6,108 KB)
 [im2.jpg](#) (5,873 KB)
 [im3.jpg](#) (5,652 KB)

O objectivo deste exercício é criar duas classes que compõem uma máquina de jogo (slot-machine) como a da imagem:



Images courtesy of <http://www.shutterstock.com/>

Na classe principal, SlotMachine, deve definir os atributos e o construtor da aplicação. Deve escrever o código que cria a moldura, um painel central para três imagens, lado a lado (assuma, se achar necessário, que as imagens têm 60 x 60 pixels). Inicialmente estas imagens devem conter um ficheiro chamado "imagem1.jpg", Deve ainda ter num painel a sul botões de iniciar e parar.

Para ter as imagens no ecrã use uma JLabel para cada uma delas e adicione as imagens do seguinte modo:

```
label.setIcon(new ImageIcon("im" + n + ".jpg"));
```

O botão de iniciar deve lançar um processo ligeiro (Wheel) associado a cada JLabel.

Os processos (Wheel) que substituem as imagens são independentes e estão continuamente a alterar a sua imagem por uma de N imagens com nomes "im1.jpg", "im2.jpg", até "imN.jpg", até serem interrompidos. Pode assumir que N é igual 3 se necessário. Os ficheiros de imagens estão em anexo.

Deve criar a(s) sentinela(s) dos botões de início e paragem e definir o comportamento ao carregar em cada um dos botões. O botão de paragem, depois de garantir que todos os processos pararam, deve chamar um método calculaPontuação() que verifica se as três imagens são iguais e nesse caso escreve "JACKPOT".



Soluções exercícios da semana