Universidade da Beira Interior

Sistemas Distribuídos

Licenciatura em Engenharia Informática

Revisões

Notas: No exercício que se segue pode omitir o tratamento de exceções.

Para ler dados do teclado, use as funções: Ler.umInt(); Ler.uma String();

1 - Suponha a classe Exemplo listada abaixo:

```
public class Exemplo {
   private static int a;
   private int b;
   public static synchronized void m1(){ …}
   public synchronized void m2() { …}
   public void m3(){ …}
   public static  void m4() ….
}
```

Suponha também um programa que declara dois objetos do tipo exemplo, o1 e o2 e lança duas Threads T1 e T2 que invocam um método nalgum dos objetos. Para cada situação esquematizada abaixo diga **justificando** se os métodos invocados são ou não executados em exclusão mútua.

Exemplo o1= new Exemplo(); Exemplo o2= new Exemplo();	T1	T2
	o1.m1();	o2.m1();
	o1.m3() ;	o2.m3();
	o1.m2();	o2.m2();
	o1.m2()	o2.m3();
	o1.m2();	o1.m2();
	o1.m4()	o1.m4()
	01.m4()	o2.m4()

2 – Suponha um servidor que faz a gestão de um programa de troca de livros. Os utilizadores podem oferecer livros, e podem levar livros. O servidor contém uma lista de livros (suponha que apenas é guardado o nome do livro num objeto do tipo ArrayList<String> e que nesta fase o programa não tem persistência de dados).

Estão disponíveis 3 opções: Oferecer livro; Levantar livro; Consultar livros existentes.
Oferecer um livro é: - dado o nome do livro pelo cliente, o servidor irá inseri-lo na lista.
Levantar um livro é: - dado o nome do livro pelo cliente, o servidor deve removê-lo da lista caso ele conste da lista.
Consultar livros é: - o cliente obter do servidor a lista dos livros existentes.

Pretende-se uma aplicação com um servidor multi-threaded em que o cliente e o servidor comunicam por Sockets TCP. Para cada ligação do cliente, este pode escolher várias opções até decidir terminar.
Para incentivar a entrega de livros é realizado um sorteio, cujo prémio é um cheque livro, entre os clientes que oferecem livros. Os clientes a participar no sorteio serão aqueles que oferecem os livros número 10, 20, 30, …Assim, cada vez que é feita uma operação de oferta cujo número de ordem é múltiplo de 10, o cliente receberá uma mensagem, "Parabéns, tem um bilhete para o sorteio". Se o número da operação de oferta não for múltiplo de 10 o ciente receberá a mensagem “Obrigada pelo livro”

- Construa as classes Servidor, Cliente e Thread que no servidor comunica com o cliente.

3 – Implementar o exercício anterior em JAVA rmi.
Agora, cada vez que é feita uma operação de oferta cujo número de ordem é múltiplo de 10, o cliente receberá um callback do servidor com a mensagem: "Parabéns, tem um bilhete para o sorteio".

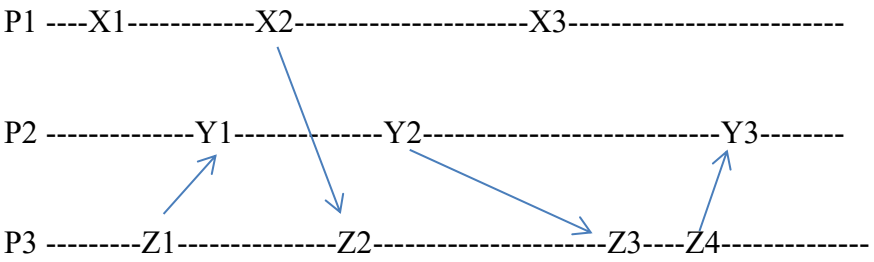
a) Construa a interface remota do objecto remoto;

b) Construa a classe que implementa o objeto remoto;

c) Construa a classe Servidor;

d) Construa o Cliente, de maneira a que possa receber o Callback do servidor, e testar cada um dos métodos do objeto remoto que executa no servidor. Não precisa de ler valores do teclado, use valores exemplo.

4 – Supondo os processos P1,P2 e P3, e usando um relógio vetorial, atribua um *timestamp* a cada um dos eventos assinalado no esquema abaixo.



Notas Auxiliares:
socket do cliente:
import java.net.*;
import java.io.*;
Socket meuCliente = null;
try { meuCliente = new Socket (“host”, portNumber); }
catch (IOException e){ … }

Obter as Streams do socket e associar objectStrems:
ObjectOutputStream os = new ObjectOutputStream (meuCliente.getOutputStream());
ObjectInputStream is = new ObjectInputStream (meuCliente.getInputStream());
String s = "exemplo";
os.writeObject(s);
s = (String) is.readObject();

socket do servidor:
ServerSocket meuServidor = null;
try { meuServidor = new ServerSocket (portNumber); }
catch (IOException e){ …}
Socket sServidor = null
try { sServidor = meuServidor.accept(); }
catch (IOException e){ }

RMI

Interface remota: java.rmi.Remote

Exceção remota: java.rmi.RemoteException

Objeto remoto : java.rmi.server.UnicastRemoteObject;

Sintaxe do nome que o objecto remoto tem no RMIregistry:
[rmi:] [//] [nomeMaquina] [:port] [/nomeObjecto]

Instalar um gestor de segurança: System.setSecurityManager (new SecurityManager());
Iniciar o registry: java.rmi.registry.LocateRegistry.createRegistry(1099);

Métodos da classe java.rmi.Naming:

void rebind (String nomeObjecto, Remote objecto);

Remote lookup (String nomeObjecto)

Class java.util.ArrayList:
ArrayList() // construtor vazio, dimensão inicial zero.
boolean **add**(Object element)
// adiciona o elemento especificado ao final da lista
void **add**(int index, Object obj)
//insere o elemento especificado na posição index
Object **remove**(int index)//remove o elemento da posição index
boolean **remove**(Object o)
//remove a primeira ocorrência do objecto dado como parâmetro
Object **set** (int position, Object obj)
// substitui o elemento da posição index pelo elemento dado
Object **get** (int position)//devolve o elemento da posição index
void **clear**() // remove todos os elementos da lista
Object **clone**() // devolve uma cópia da lista
boolean **contains**(Object element)
// devolve true se a lista contém o elemento especificado
boolean **equals** (Object obj)
// permite comparar duas listas
int **indexOf**(Object element)
// procura o índice da 1ª ocorrência de elemento
boolean **isEmpty**() // verifica se a lista não tem componentes
int **size**() // devolve a dimensão actual
String **toString** ()