OBJETOS DISTRIBUÍDOS E INVOCAÇÃO REMOTA

Trabalho em Grupo

1. O que é um *middleware* ?

Software que fornece um modelo de programação, e que é situado entre camadas de software (no nosso caso, o Java RMI, entre as camadas de aplicação e transporte), que usa um protocolo baseado em mensagens entre objetos (locais e remotos), no sentido de fornecer abstrações de mais alto nível, como as invocações e eventos remotos, proporcionando transparência de localização nas invocações remotas.

2. Quais características um *middleware* apresenta quanto à:

(a) **transparência de localização**:
O cliente não precisa saber da localização do servidor. Em RMI, o objeto que está fazendo a invocação não pode identificar se o objeto é local ou não. Nos programas distribuídos baseados em eventos, os objetos que geram eventos e os objetos que recebem notificações desses eventos não precisam conhecer a localização um dos outros.

(b) **protocolos de comunicação**:
Os protocolos que suportam as abstrações de *middleware* são independentes dos protocolos de transporte subjacentes.

(c) **hardware do computador**:
Os padrões aceitos para representação de dados externos (representação comum do CORBA, Serialização de objetos Java, XML, Referências a objetos Remotos) são usados no empacotamento e desempacotamento de mensagens. Eles ocultam as diferenças entre arquiteturas de hardware, como por exemplo a ordem do bytes nesses padrões.

(d) **sistemas operacionais**:
As abstrações de mais alto nível fornecidas pela camada de middleware são independentes dos sistemas operacionais subjacentes.

(e) **uso de linguagens de programação**:
Alguns middlewares são projetados para permitirem que os aplicativos distribuídos utilizem mais do que uma linguagem de programação.

3. O que caracteriza o modelo de objetos distribuídos ?

Interfaces Remotas : a classe de um objeto remoto implementa nos seus métodos, as operações especificadas em sua interface remota. Em Java RMI, as interfaces remotas são definidas da mesma forma que qualquer outra interface Java. Adquirem a capacidade de serem interfaces remotas estendendo uma interface como *Remote.*

Referência de objetos remotos:
A noção de referência de objeto é estendida para permitir qualquer objeto que possa receber uma RMI, tenha uma referência de objeto remoto. Essa referência é um identificador que

pode ser usado por todo um sistema distribuído para se referir a um objeto único em particular. Sua representação contém um **endereço IP**, **número de porta**, **hora**, **número do objeto local** e **interface do objeto remoto**. Juntos o número de porta e hora produzem um identificador do objeto. O objeto remoto que vai receber uma invocação num método remoto, é especificado pelo invocador como uma referência de objeto remoto. As referências de objeto remoto podem ser passadas como argumentos e resultados de invocações a métodos remotos.

4. Questões de projeto para RMI são bastante importantes na implementação de certas aplicações. Procure entender e descrever os três casos citados no Cap 5.

Escolhas para a Semântica de Invocação RMI:

Semântica de Invocação Talvez: o método remoto pode ser executado uma vez ou não ser executado.

Semântica de Invocação pelo menos uma vez: o invocador recebe um resultado quando o método remoto foi executado pelo menos uma vez, ou recebe uma exceção, informando-o que nenhum resultado foi obtido. Pode ser obtida pela retransmissão das mensagens de requisição.

Semântica de Invocação no máximo uma vez: ou o invocador recebe um resultado quando o método remoto foi executado exatamente uma vez, ou em caso contrário, uma exceção. É a semântica usada em Java RMI e CORBA, mas em CORBA é permitido a semântica talvez, usadas para métodos que não retornam resultados, e em RPC Sun é permitida a chamada pelo menos uma vez.

5. Qual a importância de transparência em RMI ?

Todas as etapas necessárias para empacotamento e troca de mensagens, mais a tarefa de localizar e contatar um objeto remoto, são ocultadas do programador que faz uma chamada remota.

6. Implementação de RMI. Descreva resumidamente o que significa cada uma dos componentes em RMI.

Objeto-Cliente:

Proxy para um objeto remoto: torna a invocação a método remoto transparente para os clientes, comportando-se como um objeto local para o invocador, mas ao invés de executar uma invocação local, ele a encaminha em uma mensagem para um objeto remoto. Oculta os detalhes da referência de objeto remoto, do empacotamento de argumentos, do desempacotamento dos resultados e do envio e recepção de mensagens do cliente. Existe um Proxy para cada objeto remoto. Um Proxy implementa os métodos da interface remota do

objeto remoto que ele representa. Cada método do Proxy empacota uma referência para o objeto alvo.

Módulo de referência remota (cliente e servidor): responsável pela transformação entre referências de objeto local e remoto e pela criação de referências de objetos remotos. Contém uma tabela de objetos remotos para registrar a correspondência entre as referências de objetos locais e as referências de objetos remotos que abrangem todo o sistema.

Módulo de Comunicação: executa o protocolo de requisição-resposta entre cliente e servidor.
Objetos Servants: são instâncias de classes que fornecem o código de um objeto remoto.

Esqueleto: a classe de um objeto remoto tem um esqueleto, o qual implementa os métodos da interface remota, mas diferente dos métodos do servant que personifica o objeto remoto. Um método de esqueleto desempacota os argumentos na mensagem de requisição e invoca o método correspondente no servant. Ele espera que a invocação termine e, em seguida, empacota o resultado, junto com exceções, na mensagem de resposta aque é enviada ao Proxy que fez a requisição.

Despachante: um servidor tem um despachante e um esqueleto para cada classe que representa um objeto remoto. O despachante recebe a mensagem de requisição do módulo de comunicação e seleciona o método apropriado no esqueleto, repassando a mensagem de requisição.

7. Qual a diferença principal entre RPC (*Remote Procedure Call*) e RMI (*Remote Method Invocation*) ? Figuras 5.7 e 5.8.

RPC implementa processos: cliente e servidor, como seria para uam linguagem procedural, como a linguagem C, que não suporta classes nem objetos.

RMI implementa objetos ao invés de processos, com nas linguagens C++ e Java.

8. Eventos e Notificações podem ser usados em uma ampla variedade de diferentes aplicações distribuídas.

a) Descreva resumidamente, o que é um **sistema distribuído baseado em eventos**.

Permitem que vários objetos em diferentes localizações sejam notificados por eventos ocorrendo em um objeto. Usam o paradigma publicar-assinar, no qual um objeto que gera eventos publica os tipos de eventos que tornará disponíveis para observação por outros objetos. Os objetos interessados em um determinado evento fazem uma assinatura para receber notificações a respeito desse evento.

b) Quais são suas principais características ?

Heterogêneos: os componentes de um sistema distribuído que não foram projetados para interagir podem trabalhar em conjunto. Basta que os objetos que geram os eventos publiquem os tipos de eventos que oferecem e que outros objetos se inscrevam nos eventos e forneçam uma interface para receber notificações.

Assíncronos: as notificações são enviadas de forma assíncrona pelos objetos geradores de eventos, para todos os objetos que fizeram uma assinatura deles.

9. Dê um exemplo de um tipo de evento e de um evento e seus atributos.

No exemplo da corretora de ações, existe um único tipo de evento: a chegada da atualização de uma ação. E os atributos podem especificar o nome da ação, seu preço corrente e a alta e a queda mais recente.

10. Quais são os participantes em um sistema distribuído de notificação de eventos ? Descreva resumidamente o que significam.

Objeto de Interesse: um objeto que sofre mudanças de estado, como resultado da invoação de seus métodos.

Evento: ocorre em um objeto de interesse como reusltado da conclusão da execução de um método.

Notificação: objeto que contém informações sobre um evento (tipo, atributos)

Assinante: objeto que se inscreveu em algum tipo de evento em outro objeto. Recebe notificações sobre esses eventos.

Objetos Observadores: objetos que desvinculam um objeto de interesse de seus assinantes.

Gerador de Eventos (Publicador): um objeto que declara que vai gerar notificações de tipos de evento particular.

11. Uma interface de *Eleição* provê duas operações:

vote: com dois parâmetros através dos quais o eleitor fornece o nome de um candidato (uma string) e o número do eleitor (um inteiro uado para garantir que cada cliente vota somente 1 vez). Os números dos eleitores são alocados, espaçadamente, dentro de um domínio de inteiros para torná-los difícil de adivinhar.

result: com dois parâmetros através dos quais o servidor fornece ao apurador o nome de um candidato e o número de votos obtidos para aquele candidato.

Discutir a semântica de invocação que pode ser utilizada quando o protocolo *request-reply* é implementado sobre uma conexão TCP/IP, que garante que os dados são entregues na ordem enviada, sem perda ou duplicação. Leve em consideração condições causando uma conexão falhar.