UNIVERSITÀ DEGLI STUDI DELLA CALABRIA
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

Esame di SISTEMI DISTRIBUITI

APPELLO DEL 31 MARZO 2004 – SOLUZIONE

Interfaccia dei Computing Server

```
import java.rmi.*;
import java.util.*;

public interface ComputingServer extends Remote
{     Vector serviziOfferti() throws RemoteException;
      Object invocaServizio(String nomeServizio, Object Parametro)
            throws RemoteException;
      void aggiungiServizio(Servizio s) throws RemoteException;
}
```

Interfaccia dei Name Server

```
import java.net.*;
import java.rmi.*;
import java.util.*;

public interface NameServer extends Remote
{     InetAddress risolvi(String nome) throws RemoteException;
      Vector elencoComputingServer() throws RemoteException;

      Vector elencoCSLivelliInferiori() throws RemoteException;
}
```

Interfaccia dei Broker

```
import java.util.*;
import java.rmi.*;

public interface Broker extends Remote
{     Vector elencoServizi(String nomeComputingServer) throws RemoteException;
      Object invoca(String nomeComputingServer, String nomeServizio,
            Object parametro) throws RemoteException;
      Object invocaGenerico(String parolaChiave, Object parametro)
            throws RemoteException;
}
```

Servizio

```
import java.util.*;
import java.io.*;

public class Servizio implements Serializable
{     private String nome;
      private String tipoInput,tipoOutput;
      private Vector descrizione;

      public Servizio(String nome, String tipoInput, String tipoOutput,
            Vector descrizione)
      {     this.nome=nome;
            this.tipoInput=tipoInput;
            this.tipoOutput=tipoOutput;
            this.descrizione=descrizione;
      }

      public String getNome()
      {     return nome;
      }

      public String getTipoInput()
      {     return tipoInput;
      }

      public String getTipoOutput()
      {     return tipoOutput;
      }

      public Vector getDescrizione()
      {     return descrizione;
      }
}
```

Implementazione dei Computing Server

```
import java.rmi.*;
import java.util.*;
import java.rmi.server.*;

public class ComputingServerImpl extends UnicastRemoteObject implements ComputingServer
{     private HashMap servizi;

      public ComputingServerImpl() throws RemoteException
      {     super();
            servizi=new HashMap();
      }

      public Vector serviziOfferti()
      {     return new Vector(servizi.keySet());
      }

      public Object invocaServizio(String nomeServizio,Object parametro)
      {     if(servizi.containsKey(nomeServizio))
                  return new Integer(nomeServizio.hashCode());
            else
                  return new Integer(-1);
      }

      public void aggiungiServizio(Servizio s)
      {     servizi.put(s.getNome(),s);
      }
}
```

Implementazione dei Broker

```
import java.util.*;
import java.rmi.*;
import java.net.*;
import java.rmi.server.*;

public class BrokerImpl extends UnicastRemoteObject implements Broker
{     InetAddress indirizzoNameServer;

      public BrokerImpl(InetAddress indirizzoNameServer) throws RemoteException
      {     super();
            this.indirizzoNameServer=indirizzoNameServer;
      }

      public Vector elencoServizi(String nomeComputingServer)
      {     ComputingServer cs=trova(nomeComputingServer);
            Vector ret=new Vector();
            try
            {     ret=cs.serviziOfferti();
            }
            catch (Exception e)
            {     System.out.println(e);
            }
            return ret;
      }

      public Object invoca(String nomeComputingServer, String nomeServizio,
            Object parametro)
      {     ComputingServer cs=trova(nomeComputingServer);
            Object ret=null;
            try
            {     ret=cs.invocaServizio(nomeServizio,parametro);
            }
            catch (Exception e)
            {     System.out.println(e);
            }
            return ret;
      }

      public Object invocaGenerico(String parolaChiave, Object parametro)
      {     Object ret=null;
            try
            {     NameServer ns = (NameServer)Naming.lookup("rmi://"+
                        indirizzoNameServer.getHostAddress()+":1099/NameServer");
                  Vector ipCSs=ns.elencoComputingServer();
                  boolean trovato=false;
                  for(int i=0;i<ipCSs.size() && !trovato;i++)
                  {     ComputingServer cs=(ComputingServer)Naming.lookup("rmi://"+
                              (InetAddress)ipCSs.get(i)+":1099/ComputingServer");
                        Vector servizi=cs.serviziOfferti();
                        for(int j=0;j<servizi.size() && !trovato;j++)
                        {     Servizio s=(Servizio)servizi.get(j);
                              if(s.getDescrizione().contains(parolaChiave))
                              {     trovato=true;
                                    ret=cs.invocaServizio(s.getNome(),parametro);
                              }
                        }
                  }
            }
            catch (Exception e)
            {     System.out.println(e);
```

```
            }
            return ret;
      }

      private ComputingServer trova(String nomeComputingServer)
      {     ComputingServer ret=null;
            try
            {     NameServer ns = (NameServer)Naming.lookup("rmi://"+
                        indirizzoNameServer.getHostAddress()+":1099/NameServer");
                  InetAddress ipCS=ns.risolvi(nomeComputingServer);
                  ret=(ComputingServer)Naming.lookup("rmi://"+
                        ipCS.getHostAddress()+":1099/ComputingServer");
            }
            catch (Exception e)
            {     System.out.println(e);
            }
            return ret;
      }
}
```

Implementazione dei Name Server (con organizzazione gerarchica)

```
import java.rmi.*;
import java.util.*;
import java.net.*;
import java.rmi.server.*;

public class NameServerImpl extends UnicastRemoteObject implements NameServer
{     private String dominio; //la radice ha dominio ""
      private InetAddress padre;
      private HashMap figli; //associazione sottodominio->InetAddress del figlio
      private boolean eFoglia;

      public NameServerImpl(String dominio,InetAddress padre,
            boolean eFoglia) throws RemoteException
      {     super();
            this.dominio=dominio;
            this.padre=padre;
            this.eFoglia=eFoglia;
      }

      public void aggiungiFiglio(String sottoDominio, InetAddress indirizzo)
      {     figli.put(sottoDominio,indirizzo);
      }

      public InetAddress risolvi(String nomeComputingServer)
      {     InetAddress ret=null;
            if(eFoglia)
            {     if (nomeComputingServer.startsWith(dominio))
                  {     String sottoDominio=nomeComputingServer.
                              substring(dominio.length());
                        String nomeFoglia=sottoDominio.
                              substring(0,nomeComputingServer.indexOf("."));
                        if(figli.containsKey(nomeFoglia))
                              ret=(InetAddress)figli.get(nomeFoglia);
                  }
                  else
                        return risolviSuAltroNodo(padre,nomeComputingServer);
            }
            else
            {     if (nomeComputingServer.startsWith(dominio))
                  {     String sottoDominio=nomeComputingServer.
                              substring(dominio.length());
                        String nomeFiglio=sottoDominio.
                              substring(0,nomeComputingServer.indexOf("."));
                        if(figli.containsKey(nomeFiglio))
                              return risolviSuAltroNodo((InetAddress)figli.
                                    get(nomeFiglio),nomeComputingServer);
                  }
                  else
                        return risolviSuAltroNodo(padre,nomeComputingServer);
            }
            return ret;
      }

      private InetAddress risolviSuAltroNodo(InetAddress indirizzoNodo,
            String nomeComputingServer)
      {     if(indirizzoNodo==null)
                  return null;

            InetAddress ret=null;
            try
```

```
            {     NameServer ns = (NameServer)Naming.lookup("rmi://"+
                        indirizzoNodo.getHostAddress()+":1099/NameServer");
                  ret=ns.risolvi(nomeComputingServer);
            }
            catch (Exception e)
            {     System.out.println(e);
            }
            return ret;
      }

      public Vector elencoComputingServer()
      {     Vector ret=new Vector();
            InetAddress radice=risolvi("");
            try
            {     NameServer ns = (NameServer)Naming.lookup("rmi://"+
                        radice+":1099/NameServer");
                  ret=ns.elencoCSLivelliInferiori();
            }
            catch (Exception e)
            {     System.out.println(e);
            }
            return ret;
      }

      public Vector elencoCSLivelliInferiori()
      {     if(eFoglia)
                  return new Vector(figli.values());

            Vector ret=new Vector();
            for(Iterator it=figli.values().iterator();it.hasNext();)
            {     InetAddress figlio=(InetAddress)it.next();
                  try
                  {     NameServer ns = (NameServer)Naming.lookup("rmi://"+
                              figlio+":1099/NameServer");
                        ret.add(ns.elencoCSLivelliInferiori());
                  }
                  catch (Exception e)
                  {     System.out.println(e);
                  }
            }
            return ret;
      }
}
```