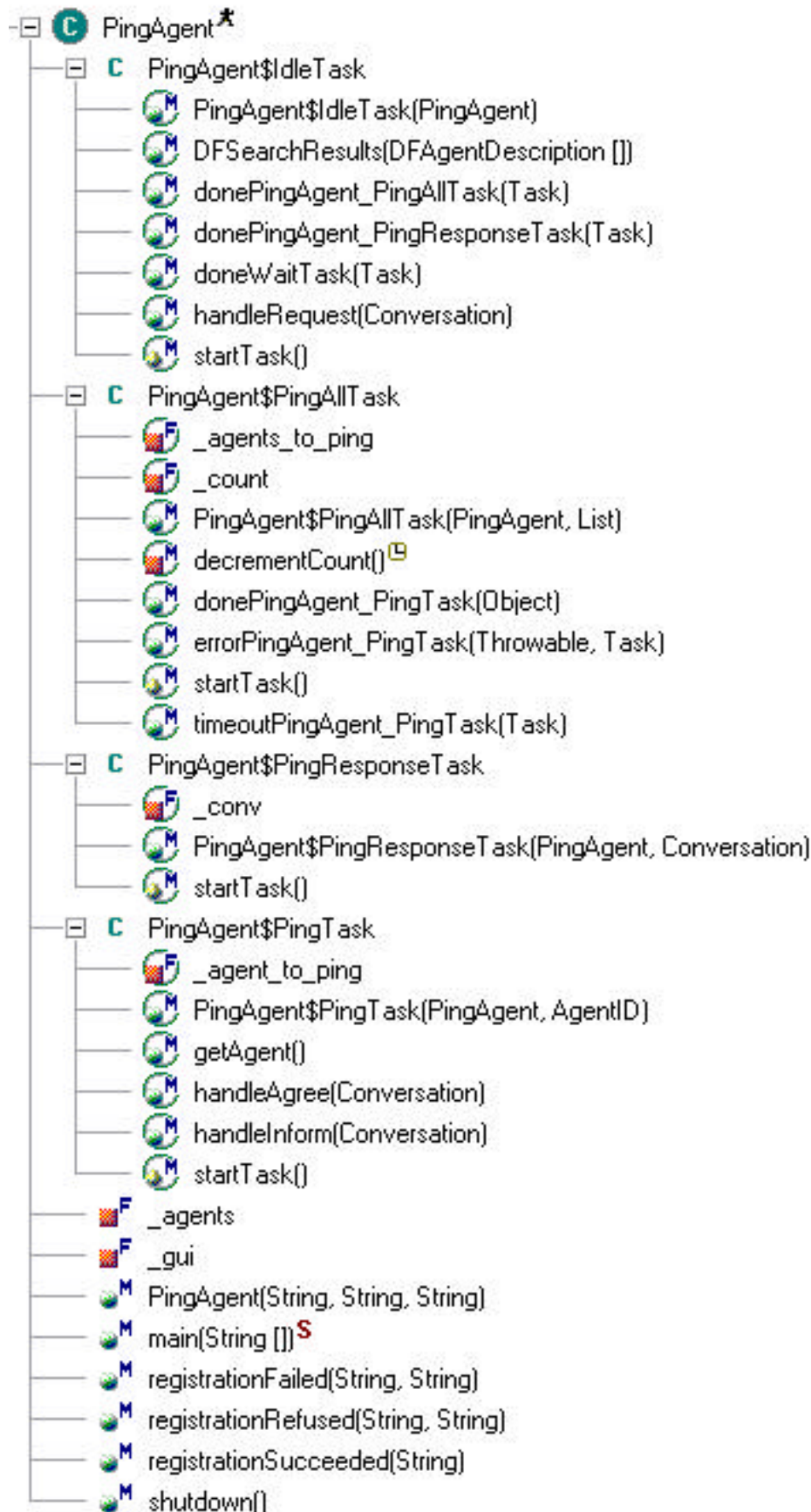
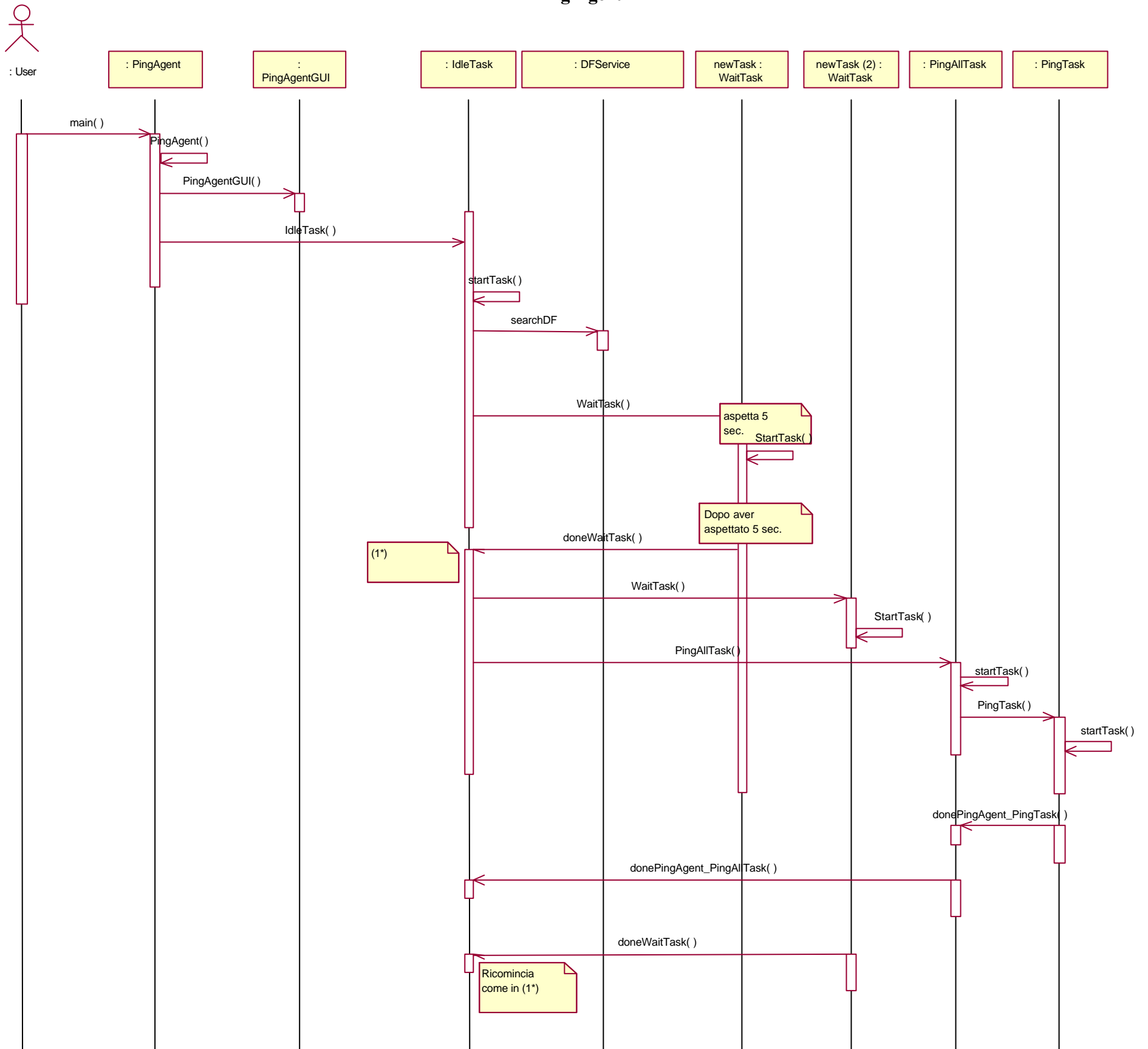


PING AGENT



Ping Agent



```

package fipaos.tutorial;

// Import the fipa-os classes
import fipaos.agent.*;
import fipaos.agent.conversation.*;
import fipaos.agent.task.*;
import fipaos.mts.*;
import fipaos.ont.fipa.*;
import fipaos.ont.fipa.fipaman.*;
import fipaos.parser.acl.ACLMessage;
import fipaos.util.DIAGNOSTICS;
import java.util.*;

public class PingAgent extends FIPAOSAgent
{
    public PingAgent( String platform, String name, String ownership )
    {
        super( platform, name, ownership );

        ...
        try
        {
            // Create a GUI for this Agent so we can see who its pinging!
            _gui = new PingAgentGUI( getAID().getName() );
        }
        catch( Throwable t )
        {
        }

        setListenerTask( new IdleTask() );

        // Send the appropriate messages to register this PingAgent with
        // the AMS and DF
        registerWithAMS();
        DIAGNOSTICS.println( "PA: Registering with the AMS", DIAGNOSTICS.LEVEL_3 );
        registerWithDF( "ping-agent" );
        DIAGNOSTICS.println( "PA: Registering with the DF", DIAGNOSTICS.LEVEL_3 );
    }
}

```

```
}
```

```
public void registrationSucceeded( String agent )  
{  
    DIAGNOSTICS.println( "PA: Registration with " + agent + " succeeded.",  
        DIAGNOSTICS.LEVEL_MAX );  
}
```

```
public void registrationFailed( String agent, String reason )  
{  
    if ( reason.compareTo( FIPAMANCONSTANTS.AGENT_ALREADY_REGISTERED ) == 0 )  
    {  
        DIAGNOSTICS.println( "PA: Already registered with " + agent + ".",  
            DIAGNOSTICS.LEVEL_MAX );  
    }  
    else  
    {  
        DIAGNOSTICS.println( "PA: Registration with " + agent + " failed (" + reason + ").",  
            DIAGNOSTICS.LEVEL_MAX );  
        shutdown();  
    }  
}
```

```
public void registrationRefused( String agent, String reason )  
{  
    if ( reason.compareTo( FIPAMANCONSTANTS.AGENT_ALREADY_REGISTERED ) == 0 )  
    {  
        DIAGNOSTICS.println( "PA: Already registered with " + agent + ".",  
            DIAGNOSTICS.LEVEL_MAX );  
    }  
    else  
    {  
        DIAGNOSTICS.println( "PA: Registration with " + agent + " refused (" + reason +  
            ").",DIAGNOSTICS.LEVEL_MAX );  
        shutdown();  
    }  
}
```

```
    }  
}  
public void shutdown()  
{  
    // Dispose of the GUI  
    if( _gui != null )  
    {  
        _gui.dispose();  
    }  
  
    // Invoked the shutdown() method of FIPAOSAgent  
    super.shutdown();  
}
```

```

***** IDLETASK *****
public class IdleTask extends Task
{
    public IdleTask()
    {
        // Don't do anything on instantiation
    }

    protected void startTask()
    {
        DFAgentDescription df_desc = new DFAgentDescription();
        ServiceDescription sd = new ServiceDescription();
        sd.setServiceName("ping-agent");
        sd.setServiceType( "ping-agent" );           // We want PingAgent agents
        df_desc.setAgentServices( new ServiceDescription[] { sd } );

        // Start a search for other Agents
        searchDF( df_desc );
        DIAGNOSTICS.println( "PA: Started DF search", DIAGNOSTICS.LEVEL_4 );

        newTask( new WaitTask( 5000 ) );
        DIAGNOSTICS.println("PA: Started WaitTask.  "+
            "When it is done we shall carry out our first ping", DIAGNOSTICS.LEVEL_4 );
    }

    public void handleRequest( Conversation conv )
    {
        // Indicate what is going on
        DIAGNOSTICS.println( "PA: Recieved request.  Starting PingResponseTask",
            DIAGNOSTICS.LEVEL_4 );
        newTask( new PingResponseTask( conv ), conv );
    }

    public void donePingAgent_PingAllTask( Task t )

```

```
{
    // We can ignore this task-type completing
}

public void donePingAgent_PingResponseTask( Task t )
{
    // We can ignore this task-type completing
}

public void DFSearchResults( DFAgentDescription desc[] )
{
    ...
}

public void doneWaitTask( Task t )
{
    newTask( new WaitTask( 5000 ) );

    DIAGNOSTICS.println("PA: doneWaitTask() has been called.\n"+
        "Lets ping the agents in our agent list.", DIAGNOSTICS.LEVEL_3 );

    synchronized( _agents )
    {
        List agents = new LinkedList( _agents );
        newTask( new PingAllTask( agents ) );
    }
}
} //end of class IdleTask
```

***** PINGALLTASK *****

```

public class PingAllTask extends Task
{
    public PingAllTask( List agents )
    {
        ...
    }

    protected void startTask()      {
        ...
        {
            ...
            newTask( new PingTask( agent ), 5000 );
        }

        if( _count <= 0 )
        {
            done();
        }
    }

    public void errorPingAgent_PingTask( Throwable th, Task t ){
        ...
    }

    public void donePingAgent_PingTask( Object result ) {
        decrementCount();
    }

    public void timeoutPingAgent_PingTask( Task t ){
        ...
    }

    private synchronized void decrementCount()  {
        ...
    }
}

```

```
    }  
} //end of class PingAllTask
```

```
***** PINGTASK *****
```

```
public class PingTask extends Task
{
    public PingTask( AgentID agent )
    {
    }

    protected void startTask()
    {
        // Create an ACLMessage and forward it to our target Agent
        ACLMessage acl = getNewConversation( "fipa-request" );
        acl.setPerformative( FIPACONSTANTS.REQUEST );
        acl.setSenderAID( _owner.getAID() );
        acl.setReceiverAID( _agent_to_ping );
        acl.setContent( "( ping )" );
        acl.setOntology( "ping" );

        // Forward via super.forward(), which binds the new Conversation with this Task
        forward( acl );

        // Indicate what is going on
        DIAGNOSTICS.println( "PA: Sent ping request to> "+_agent_to_ping,
        DIAGNOSTICS.LEVEL_4 );
    }

    public void handleAgree( Conversation conv )
    {
        // Just ignore!
    }

    public void handleInform( Conversation conv )
    {
        // Inform parent!
        done( _agent_to_ping );
    }
}
```

```
}  
  }  
  //endo of class PingTask
```

***** PINGRESPONSETASK *****

```

public class PingResponseTask extends Task
{
    public PingResponseTask( Conversation conv )
    {
        ...
    }

    protected void startTask()
    {
        // Respond to the original message

        // Send Agree
        ACL msg = _conv.getACL( _conv.getLatestMessageIndex() );
        ACL resp = new ACL( msg );
        resp.setSenderAID( _owner.getAID() );
        resp.setReceiverAID( msg.getSenderAID() );
        resp.setPerformative( FIPACONSTANTS.AGREE );

        forward( resp );

        // Send Inform
        resp = new ACL( msg );
        resp.setSenderAID( _owner.getAID() );
        resp.setReceiverAID( msg.getSenderAID() );
        resp.setPerformative( FIPACONSTANTS.INFORM );

        // Forward via super.forward()
        forward( resp );

        done();
    }
}

```

```
***** PINGAGENT.MAIN *****  
  
public static void main( String args[] )  
{  
    if( args == null || args.length < 2 )  
    {  
        System.out.println( "Usage: java fipaos.tutorial.PingAgent <platform.profile>  
<name>" );  
        return;  
    }  
  
    new PingAgent( args[0], args[1], "FIPA-OS" );  
}  
}
```