Unit 4B Java Remote Method Invocation: Further Look

Unit Outcomes. Here you will learn

- how to setup event notification in Java RMI
- how to setup and use a remote factory, why this is useful
- about the lifetime of remote objects and how to manage it
- what errors can occur during RMI and how they can be handled

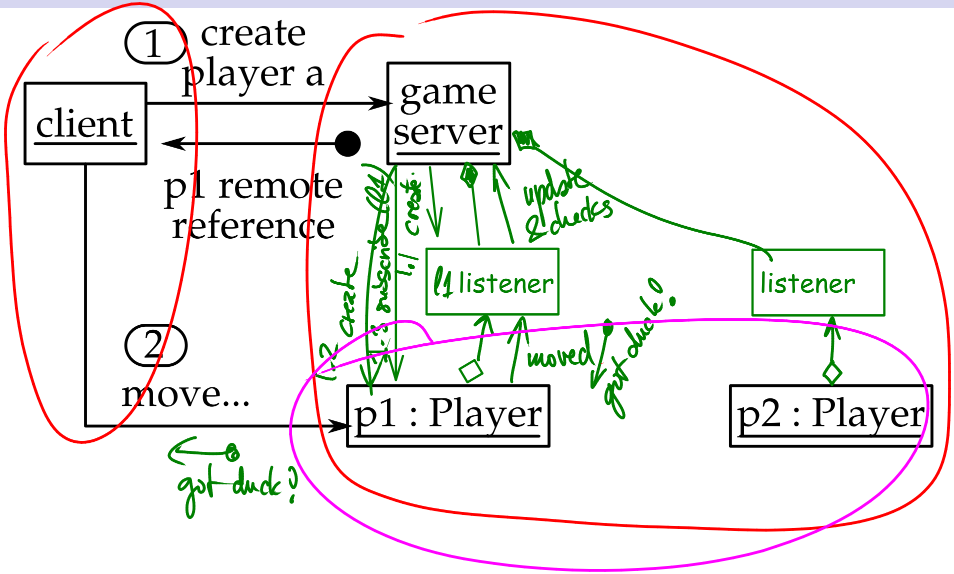
Further Reading: Sun RMI Specs + Grosso 2001 Java RMI ch17,16

Contents

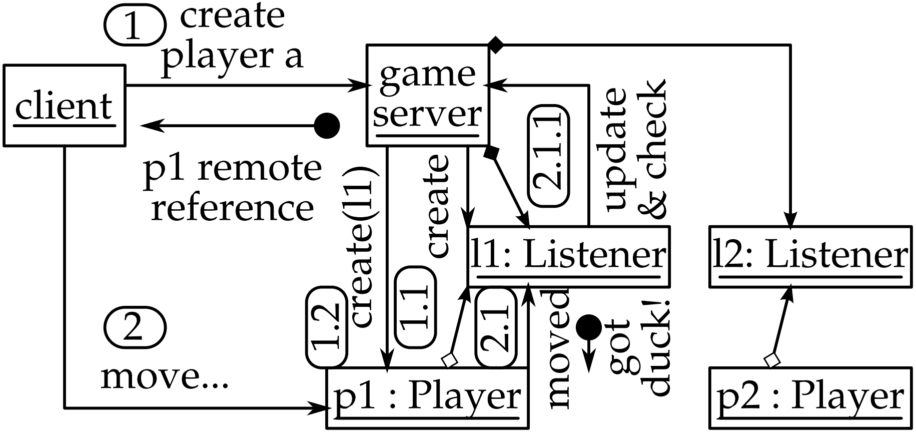
Notification

- examples:
  - chat server notifies chat clients
  - player notifies server of its movement
- 2 solutions:
  - game server = listener
  - dedicated listener:

Dedicated listeners



Dedicated listeners



Using anonymous inner class

```
private void subscribeToPlayer(final PlayerInterface player)
     throws RemoteException
{
     player.subscribe
      (
          new PositionListenerInterface()
          {
               public String newPosition(int x, int y)
                    throws RemoteException
               {
                    return playerMoved(player);
               }
          }
     );
}
```

private String playerMoved(Player p){...}

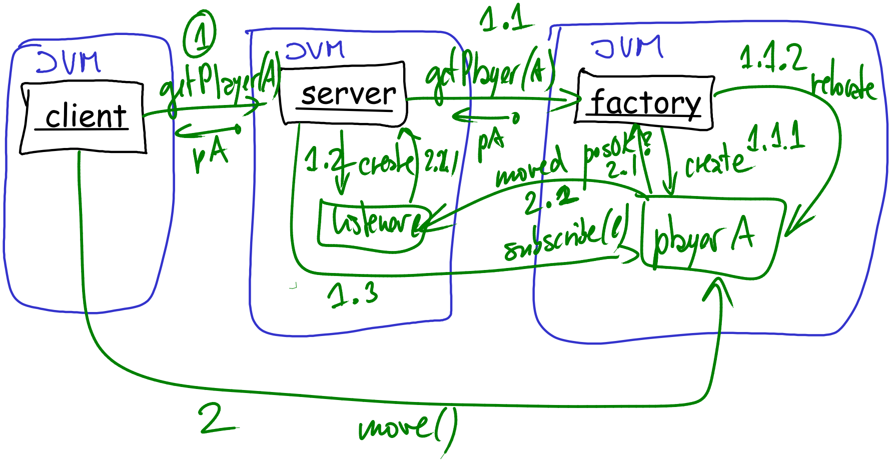
Using named inner classes

```
private void subscribeToPlayer(final PlayerInterface player)
    throws RemoteException
{
    player.subscribe(new PlayerListener(player));
}

private class PlayerListener // inner class
    extends UnicastRemoteObject
    implements PositionListenerInterface
{
    private PlayerInterface player;

    protected PlayerListener(PlayerInterface player)
        throws RemoteException { the usual body }

    public String newPosition(int x, int y)
        throws RemoteException
    {
        return playerMoved(player); // method of outer class
    }
}
```

Factories
Definition and benefits

- *factory* =
  object creating and managing instances of another class
- why?
  - *naming instances* for easier sharing, including distributed
  - awareness of *past instances*, eg when making a new one
  - can *reuse inactive instances* instead of making new ones
  - allows complete *encapsulation* of class code

Player factory



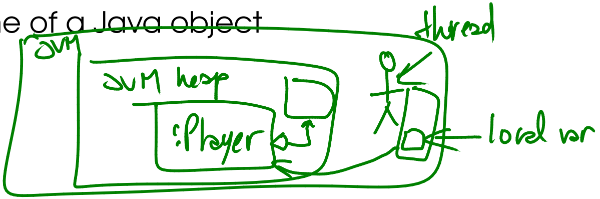
Encapsulation of player details

```
public interface PlayerFactoryInterface extends Remote
{
    PlayerInterface getPlayer(String name)
        throws RemoteException;

    void relocatePlayer(String name)
        throws RemoteException;

    void newBounds(int xMin, int yMin, int xMax, int yMax)
        throws RemoteException;

    void newColourLimit(int colLimit)
        throws RemoteException;
}
```

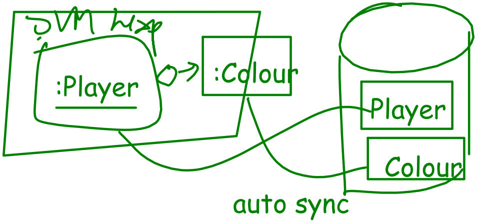
- server shares with other nodes only:
  `PlayerFactoryInterface` + `PlayerInterface` + `PositionListenerInterface` + `Direction` enum

Lifetime of remote objects
Garbage collection reminder

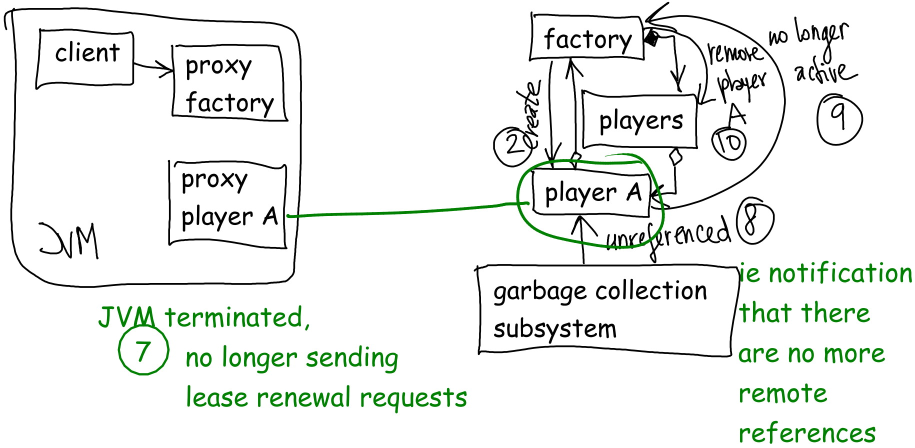
- normal lifetime of a Java object



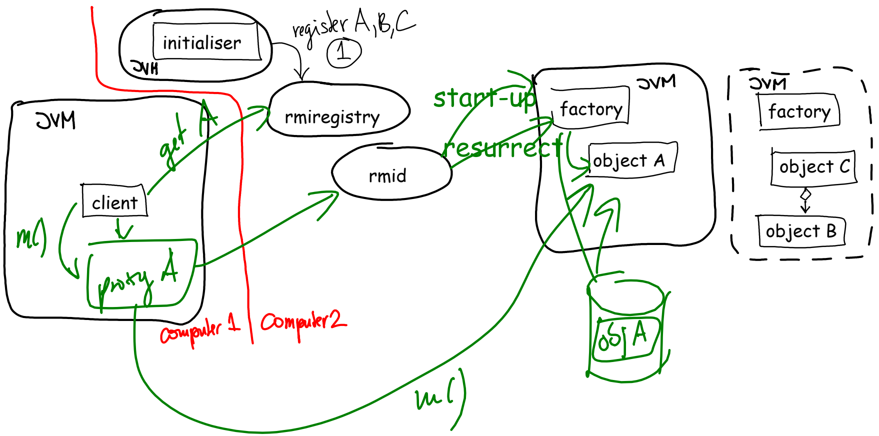
- a persistent Java object



Distributed garbage collection



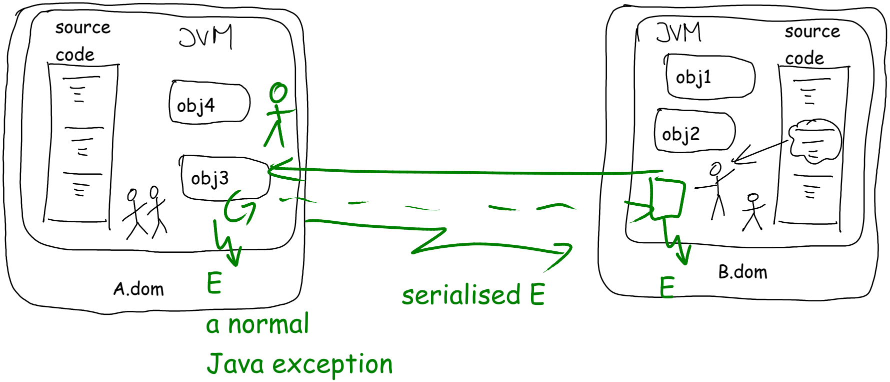
Activation framework



this is what happens when calling a method of a persistent remote object A that is no longer active in any JVM

- rmid starts a Java program that has a factory that resurrects object A

Remote exceptions
Propagating exception remotely



Overview of remote exceptions

- network configuration errors, eg:
  `java.rmi.ConnectException` (eg computer refused connection)
- network failures, eg:
  `java.rmi.ConnectIOException` (eg timeout during connect)
  `java.rmi.MarshalException` (eg timeout during data exchange)
- remote JVM crashes, updates, eg:
  `java.rmi.UnknownHostException` (eg computer renamed)
  `java.rmi.NoSuchObjectException` (eg restart, no persistence)
  `java.rmi.StubNotFoundException` (eg object no longer remote)

this slide is about errors that prevented a correct RMI;
previous slide is about an error deteceted during the remote call
and passed via a correctly functioning RMI

Learning Outcomes

Learning Outcomes. You should now be able to

- describe the purpose of notification in a distributed object model and give examples of its use
- program Java RMI notification listeners and notification subscription services
- describe the purpose of a factory using an example
- program a simple factory featuring automatic removal of unreferenced instances
- describe the Java RMI garbage collection process
- briefly describe and correctly use the Java RMI exception propagation mechanism
- list several common errors that are represented by Java RMI various remote exceptions (no need to memorise the exception names but should recognise them when shown)