JAVA RMI: CONCURRENCY CONTROL

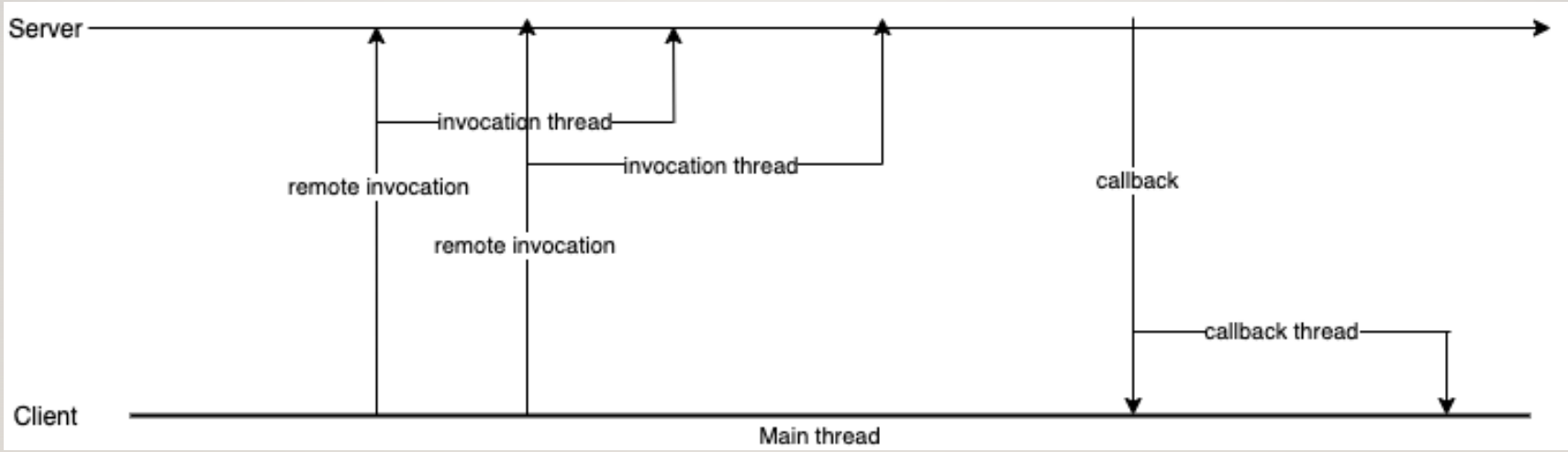
ELOI GABALDON

ELOIGABAL@GMAIL.COM

MULTI-THREADING

- RMI uses mutli-threading by default.
- Every remote method invocation is executed in a different thread.
- Several clients can contact the same server concurrently.
- We have to take care for concurrency problems and protection of single resources.

MULTI-THREADING



CONCURRENCY CONTROL

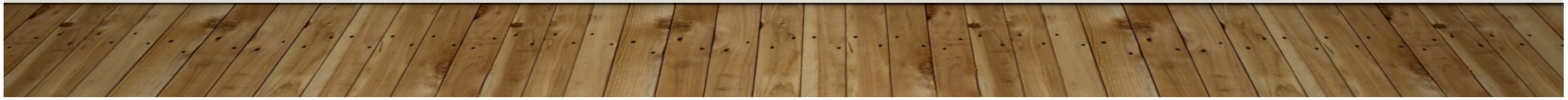
- We must implement concurrency control on the clients and the servers in order to protect the server resources and control the workflow of the application.
- We can control the applications workflow by using “synchronized” methods or blocks and also using the “wait()” and ”notify()” functions.

EXAMPLE: SECRET NUMBER MULTI PLAYER

- Server starts and selects a random number between 0-10.
- Server waits for 4 clients to register.
- When the 4 clients are connected, it send the game will start to all of them.
- Then the server waits to receive the 4 answers.
- When all answers are received, evaluates which ones found the secret number and sends back “winner” or “looser”.

JAVA RMI: OBJECT SHARING

ELOI GABALDON

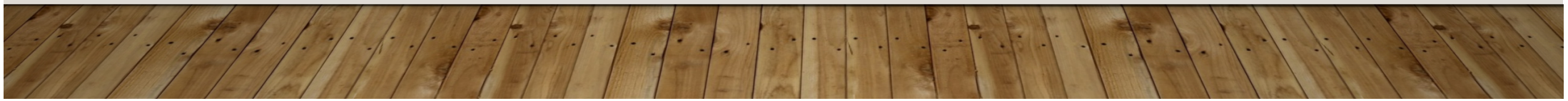


NEED TO SHARE OBJECTS

- We have seen that we can send (as parameter) and receive (as the return value) any basic types when performing a remote invocation.
- Sometimes we would like to send or receive more complex objects.
- **Implementation**
- Make the sharing object implement the "serializable" interface
- Set the shared object in the "common package"

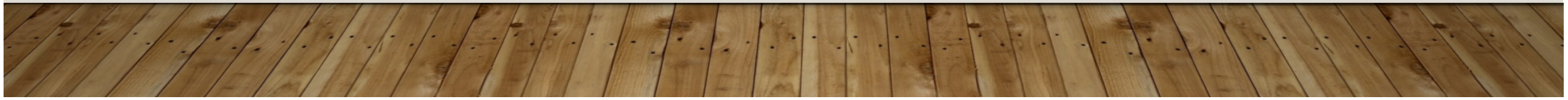
EXAMPLE: CARD DEALING GAME

- Server creates and shuffles a card deck
- Clients can ask for cards (delivered from the shuffled deck)
- Clients can play a card to the game.
- As this is an example, no game rules have been implemented.



JAVA RMI: SECURITY

ELOI GABALDON



SECURITY: INTRODUCTION

- When running java applications in a development environment, we can execute all kinds of operations without concerns. (read-write files, connect to servers, etc)
- However, it is important to secure this operations when running in a production environment.
- We can secure the operation using the SecurityManager.

SECURITY MANAGER

- You can add a security managaer in the java execution as a parameter:
  - -Djava.security.manager
- Also you will need to define a policy file
  - -Djava.security.policy=/path/to/other.policy (add the permissions to the ones found in your default system)
  - -Djava.security.policy==/path/to/other.policy (replace all the policies)

- **By default, the security manager forbids everything, and you have to give permission for each feature you are using.**

SECURITY: IN RMI

- RMI is designed to allow remote objects to be made available dynamically. This way, one component is able to download the specified code from a known location on runtime.
- This way, the components can execute code they don't know, in RMI is very important to use the security as the unknown code can be malicious.
- Actually, when running without security, the "**dynamic code downloading**" is disabled

POLICY EXAMPLE

- RMI security manager does not permit network access. Exceptions can be made via the specification in a java.policy file.

```
grant {
  // permits socket access to all common TCP ports, including the default
  // RMI registry port (1099) – need for both the client and the server.
  permission java.net.SocketPermission "*:1024-65535", "connect,accept,resolve";
  // permits socket access to port 80, the default HTTP port
  permission java.net.SocketPermission "*:80", "connect";
};
```

JAVA RMI: DINAMIC DOWNLOADING

ELOI GABALDON

DOWNLOADING UNKNOWN OBJECTS

- It is possible to create an application where the client or the server can execute different objects without knowing the real implementation of the object in the compilation time (the object is not in the common package)
- This is named dynamic downloading.
- However due to security issues (the unknown object could contain malicious software) the server and the client have to be properly set, uploading the classes to a trusted location.