RMI Feedback

Distributed Systems






Key Criteria

- **Understanding** of RMI concepts (report + code)
  - Serializable ↔ Remote ↔ Local
  - RMI Registry
- Design of distributed application
  - Distribution
  - Session management
  - Concurrency
  - Separation of Concerns
  - Interfaces
- Does it work as requested (cf. assignment)?

General impression

- **Main concepts are understood**
- **Biggest issues:**
  - **Reports** (design decisions & diagrams)
    - Terminology and writing
  - **Few "crucial" mistakes *in submitted code***

RMI CONCEPTS

Local ↔ Serializable ↔ Remote

- Main problem:
  - When should objects be Serializable / Remote / Local?
  - Not (well) addressed in design decisions
- Serializable only when necessary
  - Only to transfer data
  - By value
  - **Not:** default Serializable if not Remote
- Remote
  - Distributed services on *shared data*
  - By reference
- **Remote & Serializable: Contradictio in terminis**

Design Decisions

- Which services on different hosts & remotely accessible?
- Use RMI registry or not?
  - RMI registry ⇒ remote object resolution (nothing more!)
  - Name ⇒ address of remote objects (remote reference)
- Life-cycle management
  - Automatic or Manual (drop ref and unexport)
  - Distributed garbage collector (keep no reference)
- Remote vs local components
  - Design decisions!
- Separation of Concerns
  - Session Manager ≠ Naming server

Locate the fault

```
public IReservationSession createReservationSession(String clientName)
                                                    throws RemoteException{
        ReservationSession rSession = new
                            ReservationSession(clientName, rentalStore);

        IReservationSession stub = (IReservationSession)
                            UnicastRemoteObject.exportObject(rSession, 0);

        Registry registry = LocateRegistry.getRegistry();
        registry.bind(clientName, stub);

        return (IReservationSession) registry.lookup(clientName);
}
```

Locate the fault

```
public IReservationSession createReservationSession(String clientName)
                                                    throws RemoteException{
        ReservationSession rSession = new
                                ReservationSession(clientName, rentalStore);

        IReservationSession stub = (IReservationSession)
                                UnicastRemoteObject.exportObject(rSession, 0);

        Registry registry = LocateRegistry.getRegistry();
        registry.bind(clientName, stub);

        return (IReservationSession) registry.lookup(clientName);
}
```

Register and then lookup of same remote object.
2 (remote) calls to get something you already have...

DESIGN REPORTS

Design Reports

- To the point
  - No sequential stories!
- Most reports were consistent with the code
- **BUT**
  - Bad writing style
  - Bad textual structure
    - "why" part is often wrong/skipped/undervalued
  - UML diagrams
    - Missing annotations in class diagram (Remote, Serializable)
    - Missing connections between nodes in deployment diagram

Writing style: rule #1

Bring a message

Writing style

- Structure
  - Top-down: start with overview and then refine
  - First your design, then alternatives
- Terminology
  - Bad: “the RMI server” (vague), “registered into RMI” (registry?)
  - Good: host, node, tier, component...
- Know your audience
  - TAs know Java, RMI...
- Self-contained
  - No references to outside the text (e.g. Java EE assignment)

Writing style: rule #2

Sell the message

Writing style

- Declarative style
  - Not conditional (“we think...”, “one can...”, “if...”)
- Don’t say:
  “At the ReservationSession, it is not enforced that you receive every time the same session if you request a particular session with a specific username.”
- Do say:
  “There can be more than one session per user name.”
  or
  “User names are not unique.”

SYNCHRONIZATION

Synchronization

- **When?**
  - Multiple parallel requests are possible
  - AND when this can lead to inconsistency
  - For example:
    - Confirm quotes
    - Registering car rental companies
- **Not:**
  - Most of the data *retrieval* methods
    - createQuote → tentative reservation
    - Performance bottleneck!
  - When no parallel requests are possible

Summary

- Main concepts are understood
- Report
  - Terminology
  - Motivate your design decisions
  - Correspondence to source code
  - Writing: be precise and to the point
- RMI
  - Serializable ↔ Remote ↔ Local
- Handling concurrency
  - Synchronize only where necessary