RMI Feedback

Distributed Systems






Key Criteria

- **Understanding** of RMI concepts (report + code)
  - Serializable ⇔ Remote ⇔ Local
  - Naming service
- Design of distributed application
  - Distribution
  - Session management
  - Synchronization
  - Central reservation system (facade)
- Does it work as requested (cf. assignment)?

General impression

- **Main concepts are understood**
- **Biggest issue:**
  - **Reports** (design decisions & diagrams)
    - Explain design decisions (reasoning)
    - Language
  - 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 to transfer data (=> by value)
- Remote
  - Transfer remote reference
  - Distributed services on *shared data*
- **Not:**
  - *Default Serializable if not Remote*
- **Remote & Serializable: contradictio in terminis**

Naming service

- For distributed car rental agency
  - Small yet **crucial piece**
- Why necessary?
  - Many, distributed car rental companies (CRCs)
  - Central entry point for discovery CRCs and use
- How to realize?
  - (a) Custom class for storing/looking up (name -> remote reference) or
  - (b) Using the RMI registry
  - **MUST be remotely accessible**

DESIGN DECISIONS

Design Decisions

- Which services on different hosts & remotely accessible?
- Use RMI registry or not?
  - RMI registry is a Naming service (nothing more!)
  - Name ⇒ address of remote objects (remote reference)
- Remote vs local sessions
  - Design decisions!
  - No combined solution possible
- Life-cycle management
  - Manual (drop ref and unexport)
  - Distributed garbage collector (keep no reference)

Design Decisions (cont.)

- Stateless vs stateful sessions
  - ManagerSession is stateless
  - One instance can serve all managers!
  - Compare: @Stateful, @Stateless in JEE

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
    - Redundancy in sequence diagrams => use loops and alt structures

Writing style: rule #1

Bring a message

Writing style

- Structure
  - Guide the reader
  - Top-down: start with overview and then refine (use subsections!)
  - First your design, then alternatives
- Know your audience
  - TAs know Java, RMI...
- Read your own text
  - “are sen**d**”

Some terminology

- RMI server/The server (vague)
  - Which “server”? (central agency vs. CarRentalCompany)
  - Define what “server” refers to
    - Physical machine => hardware
    - Software component providing the service
- RMI registry/Naming service
  - RMI registry is a naming service
- Example: “register the interface of remote objects to the RMI“
  - Register the remote reference to an object to the RMI registry
- **Also at the exam!!!**

Writing style: rule #2

Focus on the message

Writing style

- Weight the pros and cons of alternatives
- Declarative style
  - Not conditional (“we think...”, “one can...”, “if...”)
- Be confident
  - “… Probably does not create a bottleneck …“

CONCURRENCY

Handling Concurrency

- **When synchronization?**
  - 1) Multiple parallel requests are possible AND
  - 2) when this can lead to inconsistency
  - For example:
    - Confirm quotes
    - Creating sessions
    - Registering car rental companies
- **But:**
  - Use fine grained locking (method <=> data structure)
- **Not:**
  - Most of the data *retrieval* methods
    - Example: createQuote (tentative reservation)
  - When no parallel requests are possible
    - Example: ReservationSession (private per-tenant session)

  → otherwise: performance overhead!

Summary

- Student **submissions:** concepts mostly **understood**
- Take-away: Understanding
  - Local ⇔ Serializable ⇔ Remote
  - Concurrency
    - **Synchronization** only where necessary!
- Take-away: Writing
  - Use correct **terminology** + be **precise** and **to the point + explain why**