15-440
Distributed Systems
Recitation 4

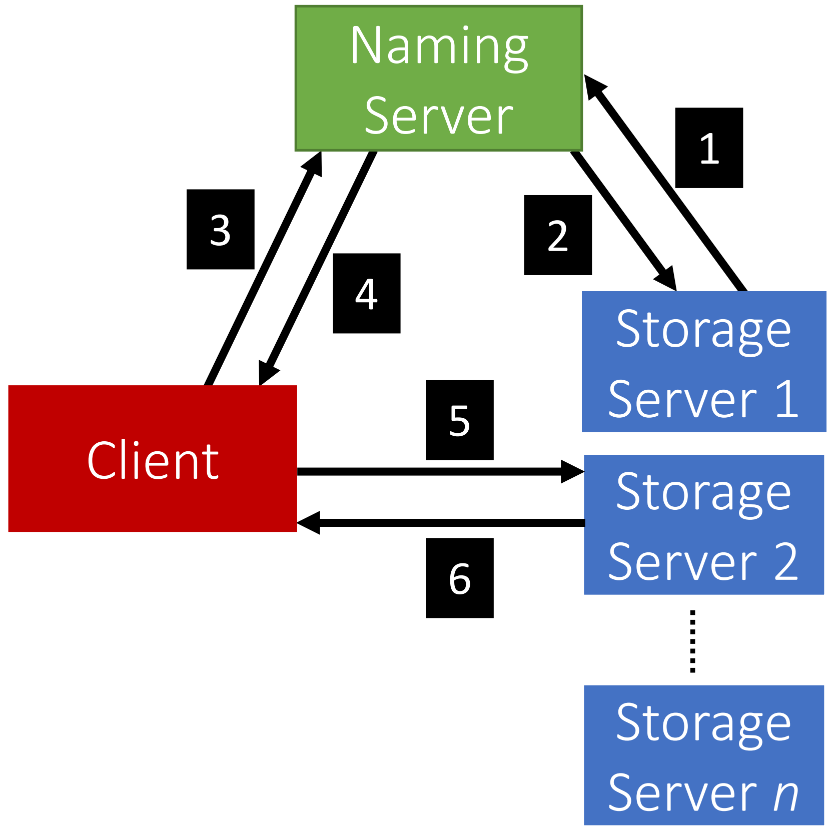
Tamim Jabban

Last Recitation

- Discussed the Entities involved and their communication
- Covered a full-fledged example that covers various stubs & skeletons
- Went over pseudocode to implement the skeleton, stub, and invocation handler

Architecture

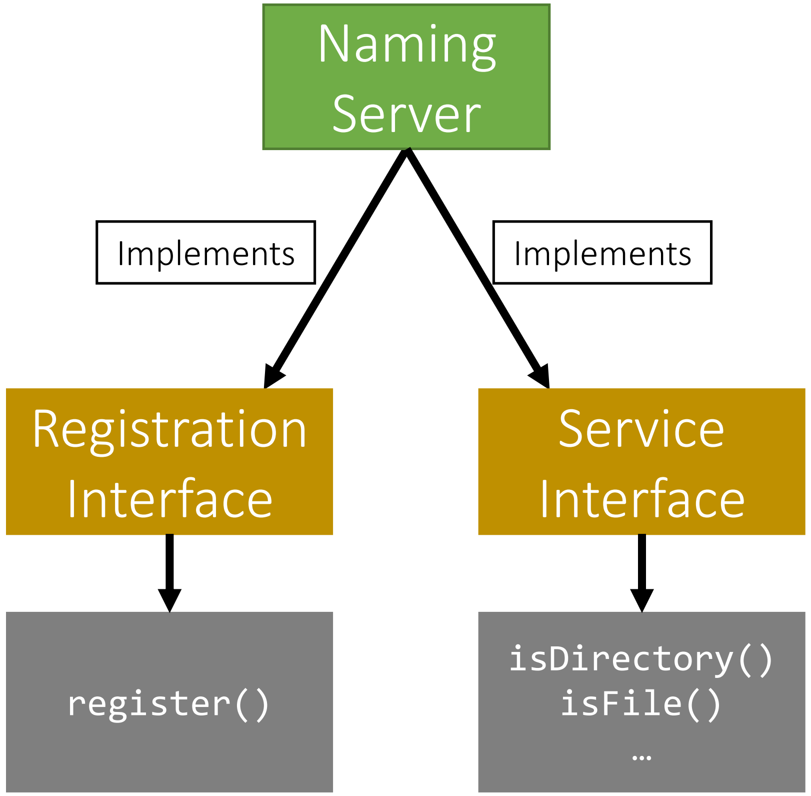
- FileStack will boast a Client-Server architecture:



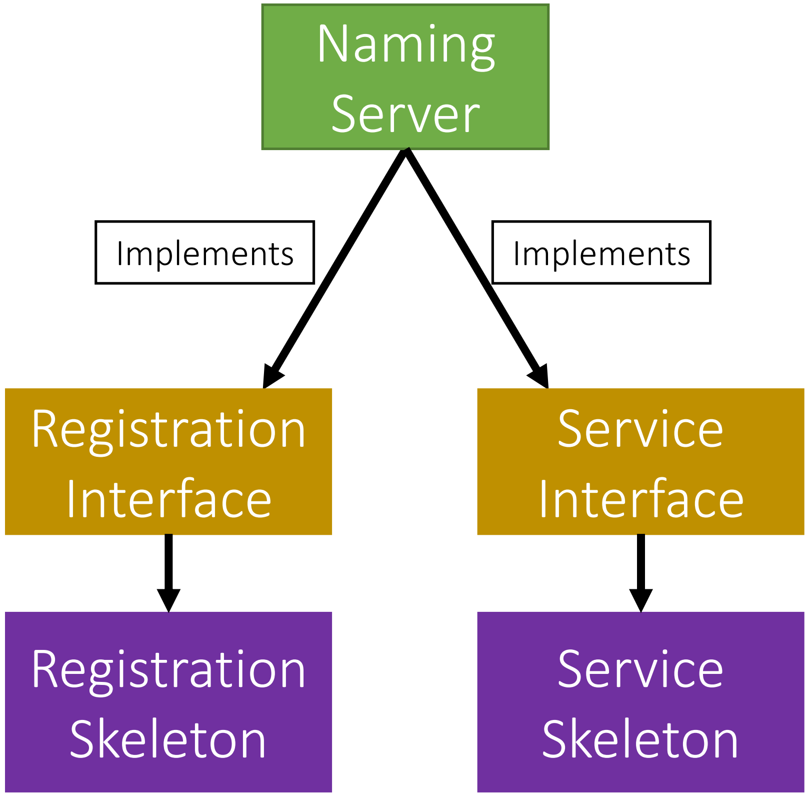
Today

- The Naming Package
- The Storage Package

The Naming Package



The Naming Package



The Naming Package

- The Naming Package:
  - Registration.java (`Interface`)
  - Service.java (`Interface`)
  - NamingServer.java (`public class`)
    - Implements:
      - Registration *Interface*
      - Service *Interface*

The Naming Package

- The Naming Package:
  - Registration.java (`Interface`)
  - Service.java (`Interface`)
  - NamingServer.java (`public class`)
    - Has Attributes:
      - Registration *Skeleton*
      - Service *Skeleton*
      - Directory Tree

Naming Package: Tree

- How can we build the *Directory Tree*?
  - One way is to use Leaf/Branch approach:
    - Leaf will represent:
      - A file and (storage) stub tuple
    - Branch will represent:
      - A list of Leafs/Branches

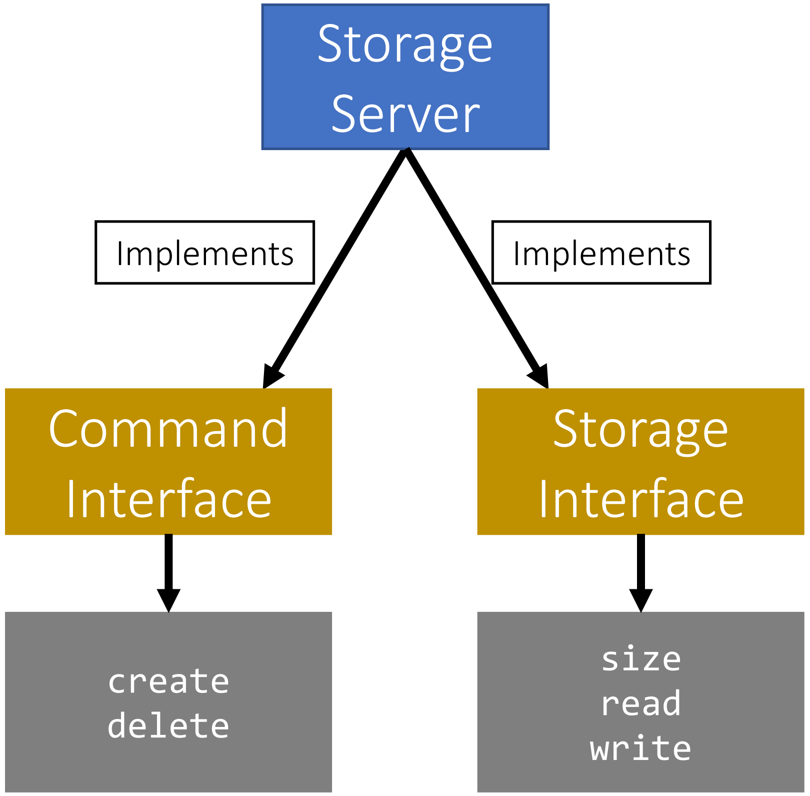
The Naming Package

- The Naming Package:
  - Registration.java (`Interface`)
  - Service.java (`Interface`)
  - NamingServer.java (`public class`)
  - NamingStubs.java (`public class`)
    - Creates:
      - Registration *Stub*
      - Service *Stub*

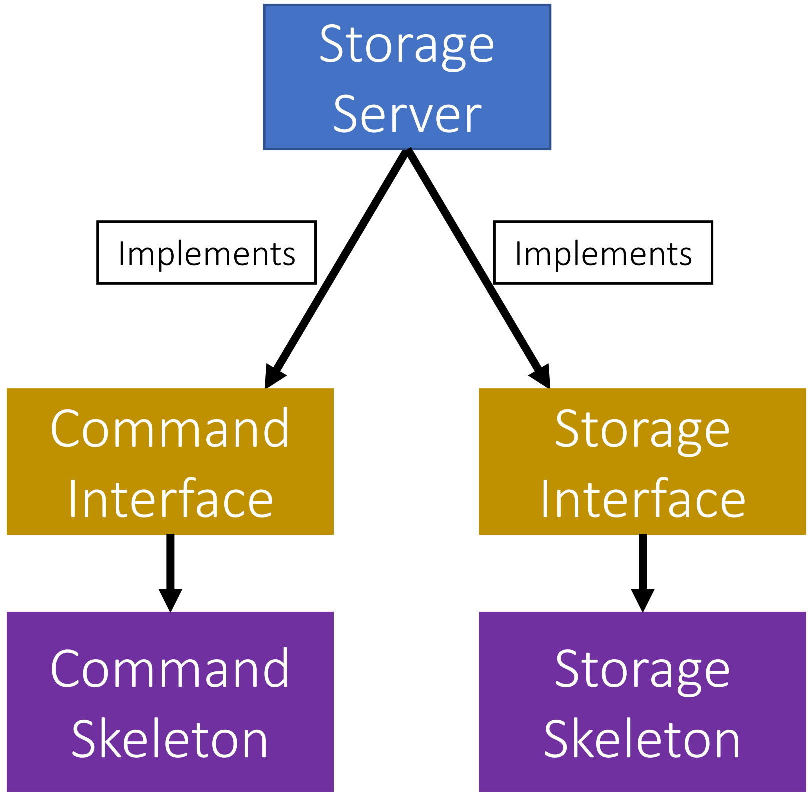
Today

- The **Naming** Package
- The Storage Package

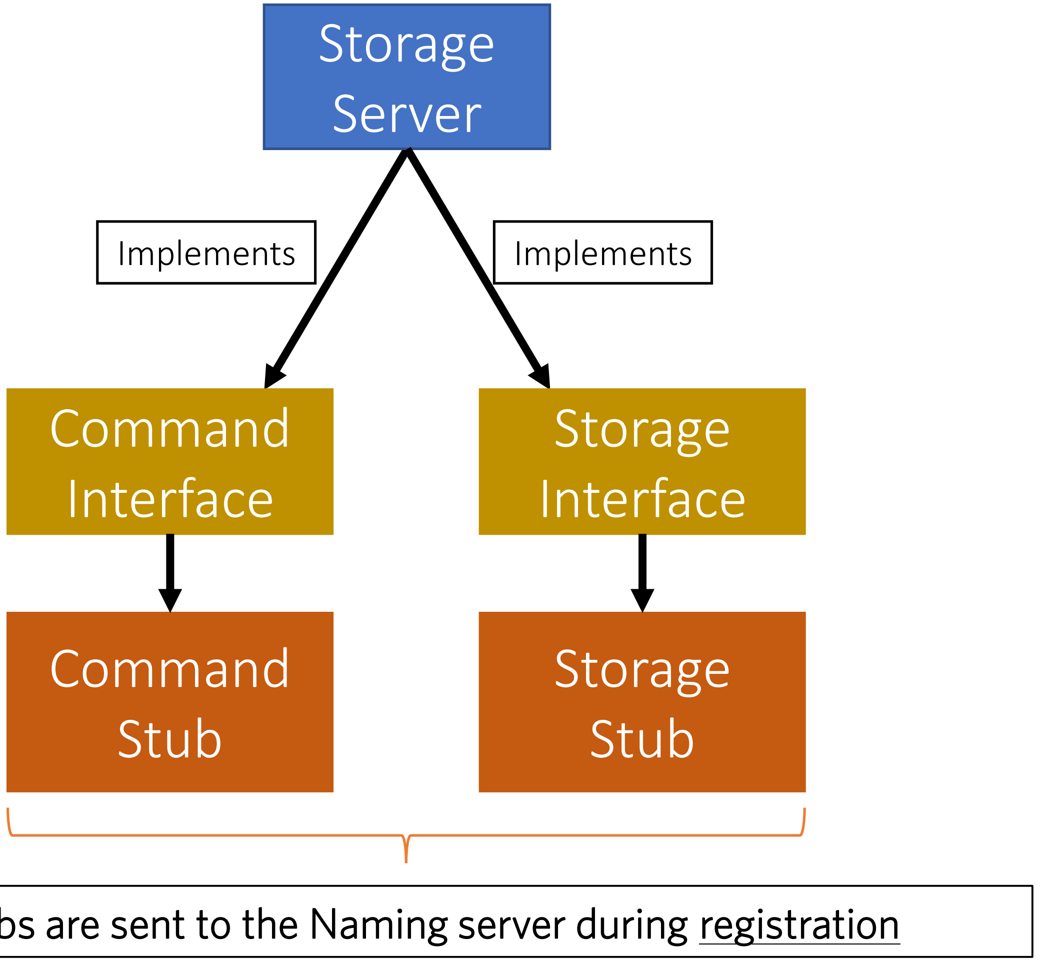
Storage Server Interfaces



The Storage Package



The Storage Package



These stubs are sent to the Naming server during registration

The Storage Package

- The Storage Package:
  - Command.java (`Interface`)
  - Storage.java (`Interface`)
  - StorageServer.java (`public class`)
    - Implements:
      - Command *Interface*
      - Storage *Interface*

The Storage Package

- The Storage Package:
  - Command.java (`Interface`)
  - Storage.java (`Interface`)
  - StorageServer.java (**`public class`**)
    - Has attributes:
      - Command *Skeleton*
      - Storage *Skeleton*
      - *Root "File"*

The Storage Package

- The Storage Package:
  - Command.java (`Interface`)
  - Storage.java (`Interface`)
  - StorageServer.java (`public class`)
    - Has functions:
      - *start()*
      - *stop()*

The Storage Package

- The StorageServer `start()` function will:
  - Start the Skeletons:
    - *Command* Skeleton
    - *Storage* Skeleton
  - Create the stubs
    - *Command* Stub
    - *Storage* Stub

The Storage Package

- The StorageServer `start()` function will:
  - **Register** itself with the Naming Server using:
    - Its **files**
    - The created **stubs**
  - Post registration, we receive a list of **duplicates** (*if any*):
    - **Delete** the duplicates
    - *Prune* **directories** if needed

The Storage Package

- The StorageServer `stop()` function will:
  - **Stop** the skeletons:
    - *Command* Skeleton
    - *Storage* Skeleton

The Design Report

- Explain the **entities** and the roles and responsibilities of each
- Project **implementation**:
  - *RMI* package
  - *Common* package
  - *Naming* package
  - *Storage* package