- Involves creating a *Distributed File System* (DFS):

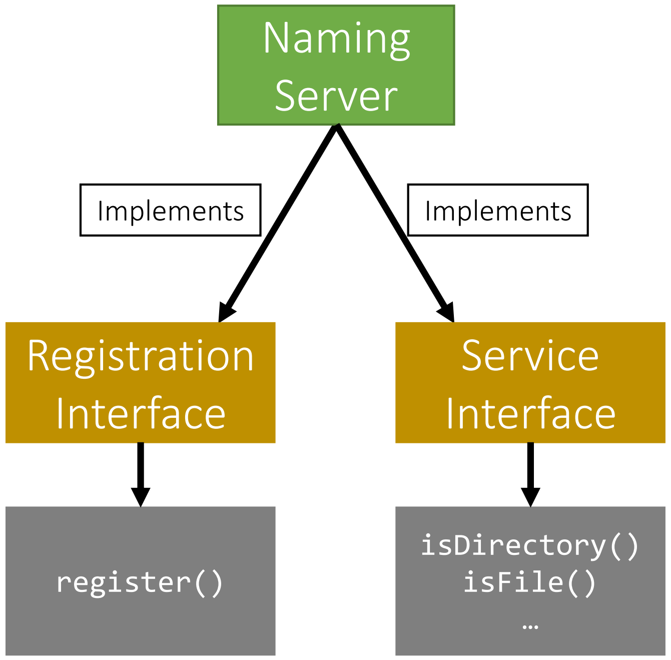
- Stores data that does not fit on a single machine
- Enables clients to perform operations on files stored on **remote servers** (RMI)

- Discussed the Entities involved and their communication
- Covered a full-fledged example that covers various stubs & skeletons
- RMI: covered Stub & Skeleton pseudocode

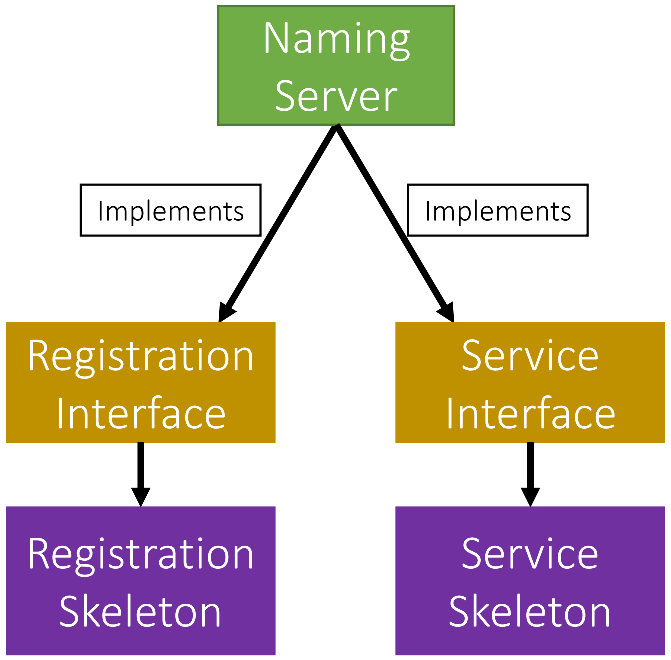
Today

- 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*
        - methods(s): register
      - Service *Interface*
        - methods(s): isDirectory, list, createFile, createDirectory, delete *(bonus)*)

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 (name) and stub
    - Branch will represent:
      - A list of Leafs/Branches

Naming Package: Classes

```
public class Node {
       String name;
}

public class Branch extends Node {
       ArrayList<Node> list;
}

public class Leaf extends Node {
       Command c;
       Storage s;
}
```

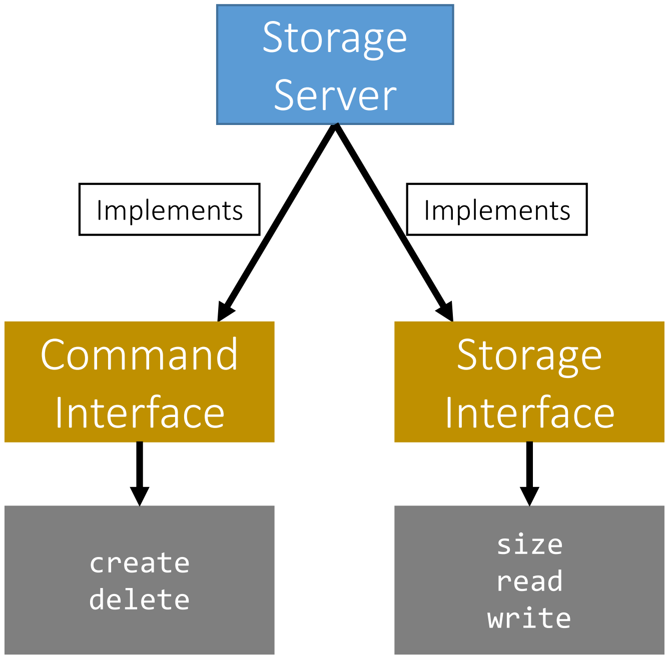
The Naming Package

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

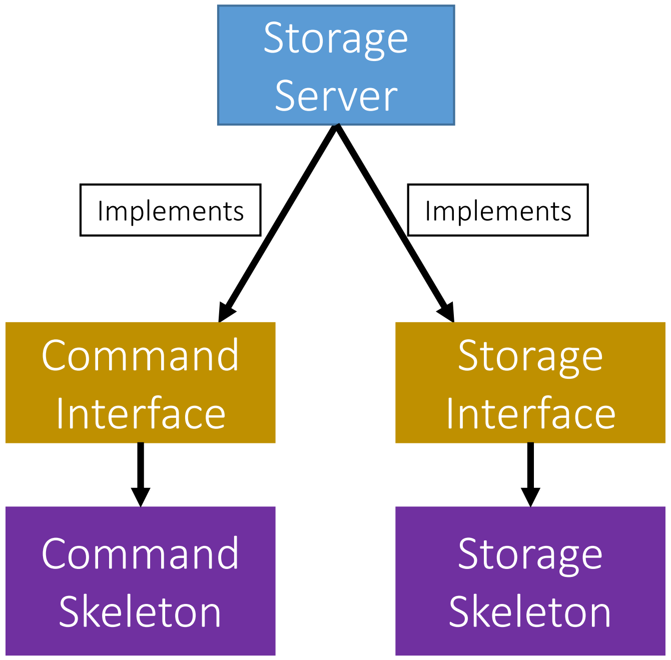
Today

- The Naming Package
- The Storage Package

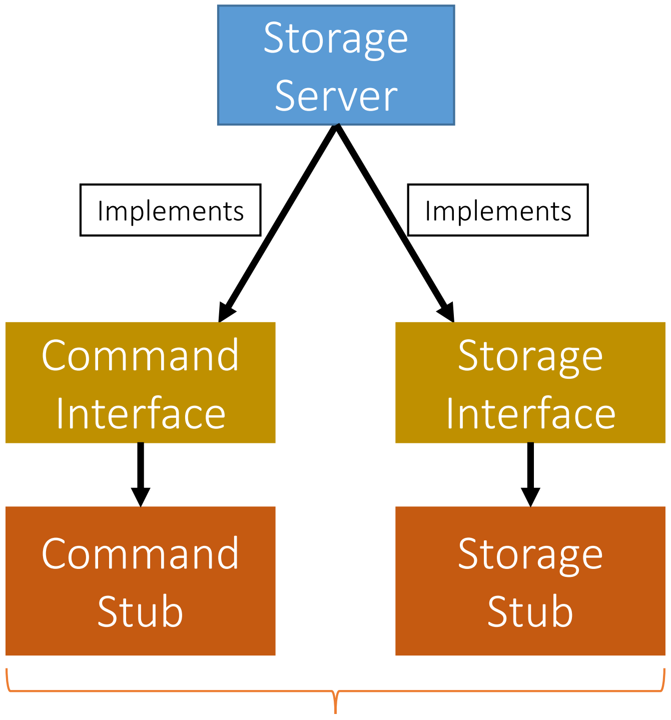
The Storage Package



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*
        - methods(s): create, delete
      - Storage *Interface*
        - methods(s): size, read, write

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:
  - **Registers** 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