② ClientCallback.java (Interface)

import java.rmi.Remote;
↳ to identify interfaces whose methods may be invoked from a non-local virtual machine.

· RemoteException.
↳ for communication-related exceptions that may occur during the execution of remote method call.

⇒ server calls the client to check it has cache data of a specific file or not.
returns the client Id.
void invalidateCache ....

FileServerService (Interface).
↳ contains methods for client to communicate with server.

Public interface FileServer ....
① void register (ClientCallback client) ....
↳ need to use clientcallback while communicating with client through server.
↳ this will return the client Id and all the cached files that client have.

(2) void unregister ....

↳ when client exits from cmd it will unregister that client from server so that server no longer tracks it.

(3) Set<String> listFiles() ....

↳ returns set of names of all files currently being hosted on the file system.

(4) void createFile (.....

↳ creates a new file on the server. if file with same name already exists then will ask for diff. name.

(5) byte[] openfile .....

↳ ~~used~~ for reading a file from the server.

1. if file does not exist → give error msg.
2. if file is there → search singleton FileSystem and return file as a byte array.
3. Register the client in cacheManager (that is this client is currently reading this file)

(6) void removeFile .....

↳ delete file from server.

↳ filesystem will invalidate cache of all the clients who had that file.

⑦ void editFile(...)
↳ edit the file on the server.
↳ filesystem will invalidate client cache which contains old file

⑧ void renameFile(...)
↳ rename a file on the server.
filesys. will invalidate client cache with old name.

Client.java.
→ all operations that a client can perform.
→ implements ClientCallback

Private final Map<String, File> fileMap = new...
↳ to store the files cached by the client.

Public void invalidateCache.....
↳ it will fetch the file using filename from the fileMap (hashMap)
↳ set the file as invalid.

Public void printFileNames.....
↳ lists all the files available on server

Public void printFile.....
↳ Prints the data of the file

? Packages.

Public File getCachedFile ...

↳ from the local file cache returns the cached file if it exists and is valid.

Public void cachefile ....

↳ caches a file in local client cache

1. if file is already there, then updates the data and mark it as valid.
2. if file does not exists, then creates a new file object and stores the data.

ClientApplication. Jewa.

↳ entry point into client application.

→ Help → print the msg.

Public static void main ...

↳ takes inputs for client port no.

↳ register the client to serverservice registry and client will be started.

↳ switch case for diff commands

help

ls - client1. printFileNames (serverService. listFiles ( ))

Object of class FileServerService

open :

final File cachedFile = client1.getCachedFile(fileName);

if ...

↳ if cached file is valid, then print it

else ...

↳ if file is not cached on client side, we need to take file from server

1. get the file from server
2. cache file for future use
3. print the file received from server

rm :

serverservice.removeFile(fileName)

↳ send remove request to server.

↳ server will remove the file and if file is cached on client side then it will also invalidate that file.

create :

serverservice.createFile(client1, fileName, contents)

client1.cacheFile(fileName, contents);

↳ sends request to server for file creation

↳ if file is successfuly created, than cache it in local client side.

modify :

serverService.editFile.....

↳ send edit req. to server. if file is cached on client side then it will be invalidated by the server.

rename :

serverService.renameFile ....

↳ send req. to server. if file is already cached on client side, it will be invalidated by server.

exit : unregister the client, break the loop, exit.

ServersService.unregister(client1);

keepGoing = False

— x — x — x — x — x —

File.java

data, isValid.

↳ getter & setter methods for both variables.

— x — x — x — x — x —

CacheException.java.

- for custom exception thrown by ClientCache-Manager.java.

— x — x — x — x — x —

Client.java.

↳ This is a server side representation of client.

Public boolean equals . . .

Public ~~boo~~ synchronized void sendCacheInvalidationEvent. . . .

try {

this.callback.invalidateCache . . .

↳ ~~see~~ invalidate given file for given client.

Private boolean stringsAreEqual . . . .

↳ string comparison

Public ClientCallback getCallback ( )

↳ returns the callback associated with this client.

Public boolean isActive ( )

↳ returns the status of client active or inactive

Public void deactivateClient ( )

↳ makes client inactive (when client types exit)

ClientCacheManager.java

→ manages relation of files to clients.

main Two tasks : 1. keep track of which client have which files cached locally.

2. sends cache invalidation notices to all clients when a file changes in the file system.

clientCacheMap → map of file and clients list which have file cached locally.

registeredClientMap → map list of registered client Id and the client object.

public static ClientCacheManager getInstance()

↳ returns the instance of clientCacheManager

public void registeredFile....

↳ stores the list of client with file name when client caches the file locally for first time.

public void sendCacheInvalidationEventToAllClients....

↳ get list of clients from clientCacheMap using filename

↳ loop over all the clients and send notification for invalidation if client is active

public void registerClient ...
↳ registers a client with the server.

public syn.. void unregisterClient ....
↳ unregisters a client with server. when it exits

File.java
↳ represents a file in a file system.

Public File ...
↳ creates a new file with given data.

Public byte[] read ....
↳ returns the content of ~~this~~ file
using lock & unlock

Public boolean modify ....
↳ modifies file using lock & unlock

public void delete () ...
↳ same → deletes files using lock on write operation

FileException.java.

↳ used for custom exception thrown by File.java and FileSystem.java

---

FileServer.java

↳ entry point into the server for operations from a client

Files.walkFileTree ----

Public void register...

↳ register a client with server

public void unregister...

↳ unregisters a client from server

Public set<String> listFiles()...

↳ generates a list of all available files

Public void createFile...

↳ creates a new file

↳ starts keeping record of where file is cached

public byte[] openfile
↳ returns the file data
if not already cached then locally stores it.

public void remove--
↳ delete file and set all its cached version as invalid.

public void editFile---
↳ edit file and make all cached version invalid

Public void renamefile---
↳ rename the file and make all cached instances invalid

FileSystem.java
↳ class for six file system operations
1. create 2. Delete 3. Modify 4. rename
5. Read 6. List.

↳ all methods are same as described previously.

ServerApplication.java

- starting point of server app.
  only used to instantiate FileServer and bind it to the file service name.