Distributed Systems
COMP90015 2020 SM2
Project 3 - Collaborative Whiteboard

Introduction

- This project is worth 20% and is to be done in a group. There is no written report, it is entirely a software project.
- You are required to implement a `WhiteboardServer` and `WhiteboardPeer` that will operate in much the same way as the server and peer for project 2b. Specific instructions are given in these slides.
- Some skeleton code has been provided as `pb3.tar.gz`. You must use the event framework as used in project 2 to implement your distributed system.
- You should work in your existing git repository, in a branch for project3 and then merge this into the master branch when you are ready to submit.

WhiteboardServer

- Some skeleton code has been provided in `WhiteboardServer.java`. You should modify this file to implement your server for this project.
- The events used by the server have been defined for you, as has some initial code for reading parameters from the command line.
- The purpose of the whiteboard server is to allow whiteboard peers to advertise themselves and their whiteboards that they are sharing, and to share and unshare those boards as desired. Therefore it uses the following events:
  - `SHARE_BOARD`: emitted by a peer to tell the server it wants to share a board
  - `UNSHARE_BOARD`: emitted by a peer to tell the server it wants to unshare a board
  - `SHARING_BOARD`: emitted by the server to tell a peer that a board is being shared
  - `UNSHARING_BOARD`: emitted by the server to tell a peer that a board is being unshared
  - `ERROR`: emitted by the server to tell a peer that an error has occurred
- The basic operation of the whiteboard server is: for every peer that connects tell the peer all boards that are currently being shared, and allow the peer to share and unshare boards, telling all connected peers whenever a peer shares or unshares a board.

Event argument format

All events require an argument. The format of the argument is given in the skeleton code. Almost all arguments follow the format:

```
peer:port:boardid%version%PATHS
```

`:` and `%` are used as delimiters, `peer` is an IP address, `port` is a numeric port number, `boardid` is a string (a timestamp as you will see in the skeleton code), `version` is a numeric number that represents the board version, and `PATHS` contains zero or more `PATH` strings:

```
PATH%PATH%....
```

`PATH` contains a `COLOR` which is `black` or `red` and 2 or more `POINT`s:

```
COLOR>POINT>POINT>....
```

A `POINT` has the format:

```
X,Y
```

`X` and `Y` are numbers.

The skeleton code provides all of the methods required to convert to and from strings for the data types used in this project.

WhiteboardPeer

- Some skeleton code has been provided in `WhiteboardPeer.java` (which probably does not need to be changed) and then in the package `pb.app`, which contains a number of classes. The only two classes in the package that you will likely need to change are `WhiteboardApp.java` and `Whiteboard.java`. You should modify these files to implement your peer for this project.
- The events used by the peer have been defined for you, as has some initial code for reading parameters from the command line, providing a GUI for the user, and some other code for manipulating data, etc.
- The purpose of the peer is to allow the user to create and modify any number of whiteboards, and to allow the user to selectively share them with other peers.

Peer operation

- Users are free to create any number of *local* whiteboards using the GUI (code is supplied and does this).
- For each locally created board, user can select `Shared` and the peer will tell the server that the board is shared (share toggle is provided in GUI).
- For each locally created board, the user can unselect `Shared` and the peer will tell the server that the board is unshared.
- The peer will list for selection (list provided in GUI) any boards that the server says are shared (including those that are not local).
- The peer will remove from selection any boards that the server says that are unshared, only if those boards are not local boards.
- The user may attempt to draw a path on any board listed for selection.
- The user may attempt to clear any board listed for selection.
- The user may attempt to undo the last path for any board listed for selection.
- The above three operations are called updates. An update is only successful if it applies to a board that has the same version as the board that the user was updating.
- For any board that is shared and not local, called a remote board, the peer will obtain the board details from the remote peer that is sharing it and will listen for updates.

- Any update by a user that should apply to a remote board will be sent to the remote peer.
- Any update received from a peer for a local board may or may not be successful, as above.
- All successful updates for a local board are sent to any peers that are listening to that board.
- When a user deletes a board, the board is removed from selection and if that board is a local board then all peers listening to it are told the board is deleted.
- When a peer is told that a board is deleted, then the board is removed from the selection.
- When a peer shuts down, i.e. the user closes the GUI, all local boards are unshared and deleted before shutdown.

Whiteboard peer events

The whiteboard peer uses the following events:

- `BOARD_LISTEN`: emitted by the peer to another peer to tell it that it wants to receive updates about the board
- `BOARD_UNLISTEN`: emitted by the peer to another peer to tell it that it no longer wants to receive updates about the board
- `GET_BOARD_DATA`: emitted by the peer to another peer to tell it that it wants to receive the entire board data for a given board
- `BOARD_DATA`: emitted by the peer to another peer to tell it the entire board data for a given board
- `BOARD_PATH_UPDATE`: emitted by the peer to another peer to request that a path be added to a given board
- `BOARD_PATH_ACCEPTED`: emitted by the peer to another peer to indicate that a path has been added to a given board
- `BOARD_UNDO_UPDATE`: emitted by the peer to another peer to request that the last path be removed from a given board
- `BOARD_UNDO_ACCEPTED`: emitted by the peer to another peer to indicate that a the last path has been removed for a given board
- `BOARD_CLEAR_UPDATE`: emitted by the peer to another peer to request that a board be cleared

- `BOARD_CLEAR_ACCEPTED`: emitted by the peer to another peer to indicate that a board has been cleared
- `BOARD_DELETED`: emitted by the peer to another peer to indicate that a board has been deleted
- `BOARD_ERROR`: emitted by the peer to another peer to indicate an error has occurred

Technical aspects

- Requires Java 11. Do not modify the `pom.xml` file.
- Your program must compile using the maven compiler, to produce a JAR in the target directory, exactly as the provided source does.
- You may modify any of the files indicated, in any way that you require, but your solution must make use of the framework provided and you may not use any other libraries, etc.

Submission

There is no written submission for Project 3.

Assessment will be based on the correctness of your submission. Submissions that fail to compile or that do not run will receive less than half marks. Submissions that partially work will receive part marks. Submissions that work as required will receive full marks.

The deadline for Project 3 is **Sunday 8th November, 23:59**.