**Midterm project: DragonSMS**

**Groups of 2**

**Problem and Specification**

This will use the Dragon example code used in the DragonREST game. In the REST version the state management is handled in the client application (i.e. the room and gamestate is passed every time you issue a command). In this version you will create a *session-based* version, the current room and gamestate will be stored in the application side and passed as needed to the RoomCommandManager (which will be supplied in a JAR along with the room classes).

The input will be supplied as text to simulate SMS. You should be able to process text with inconsistent case formatting (ALL CAPS, lowercase, miXed case, mixed spaces). Text will be processed either through a text file with several commands fed into the processor one line at a time, or via a "command-line" console input (type one command, get an output). The selected mode must be specified in the as a command line argument.

**Basic SMS Message formats**

|  |  |  |  |
| --- | --- | --- | --- |
| **FORMAT** | **REPLY** | **APP NOTES** | **MISC NOTES** |
| REGISTER <NAME> | Hello <NAME>, welcome to DragonSMS | * Store name in session |  |
| START | <Room1 intro> | * Starts new session with name * Reset gameState and Room in session |  |
| GO <ROOM#> | <Room# intro> | * Set Room in session to supplied room * Properly format the room string so it will be properly handled * Add room error check if non-existent with appropriate message |  |
| <COMMAND> <PARAMS> | <Reply based on room and gameState> | * Invoke the supplied command (with the params as needed) with current room and session * Returned gameState should be stored in session |  |
| HINT | <Reply based on room and gameState> | * Return the list of available commands given a room and game state | New functionality |

**Part I: Proposal**

Define an annotation-based framework to help solve problems of the type above. Specify what technologies you will use and how each technology will help solve the above. Specify where in your solution the technologies will be placed. Define how the annotations are to be used in the solution classes. Specify what scenarios for change the framework can handle easily and give examples.

Create UML diagrams and Interaction diagrams to show how the components are going to interact with each other. You will have two sets of class diagrams: one for the framework and one for the solution (the project using the framework)

Your proposal should have the following

* Specification of the annotation API
* List of technologies used and explanation of why you chose to use them and what benefit they bring
* UML class and interaction diagrams for the framework and solution classes
* Code examples of how the annotations will be used by the solution classes (e.g. annotating POJOs or annotating interfaces)

**Part II: Implementation**

Your implementation will be composed of 2 projects: one for the framework and one for the solution. Your solution should be decoupled from the framework (i.e. if you JAR your framework project and add it to the solution project build path it should still work)

**Bonus:**

* Add a database component using Spring JPA that will allow storing the session information, i.e. restarting the app should preserve the session and continue.
* Multiuser support -- assume the name is the unique identifier for each session, REGISTER <NAME> will switch to an existing session if present or make a new one if not