|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Original Code Modification** | **Design Concepts** | **Recommended Code Modification** |
| Remove code duplication | In Game, printing room exit code is in printWelcome() & goRoom() | - Code duplication  - Symptom of bad cohesion | Refactoring code into printLocationInfo() method |
| Add another direction of movement to North, East, South, West: Up/Down (for 1st floor, 2nd floor, basement, etc.) | Requires changes to:  - Game.goRoom()  - Game.printLocationInfo()  - Room needs 2 more fields  - Room.setExits() needs 2 more parameters  - Game.createRooms needs to be updated | - Difficult to find code, proposed fix is scattered across multiple methods & classes  - Symptom of tight coupling  - public fields mean poor encapsulation that enables tight coupling | - Decouple Game & Room classes by using a private HashMap to store exits instead of public fields  - Change Room.setExits(N,E,S,W) to Room.setExit(direction, room)  - Add getExit() method  - Still need to change Game, but future exit direction changes will be localized to the Room class (increases cohesion of Room class)  - Where does code in printLocationInfo belong? In Game or Room? Create Room.getFullDescription()  - Now in Game.goRoom(), getting the  nextRoom = currentRoom.getExit(direction);  - Now code setting exits in Game.createRooms() is easier to understand  - Option: use ENUM for exit direction rather than String |
| Now that we can go up & down, add a new room: cellar | Requires changes to:  - Game.goRoom()  - Game.printLocationInfo() - Game.createRooms()  - add public field in Room  - Room.setExits() | Loose coupling makes code modifications easier & more localized | - Game.createRooms() |
| Add an additional command word, “look” that prints out the description & exits of the current room | Requires changes to:  - validCommands field in CommandWords  - Game.processCommand()  - Game.printHelp() | - processCommand() & validCommands are tightly coupled  - Following RDD, CommandWords printing should be in CommandWords | - add command to CommandWords.validCommands  - add else if in Game.processCommand()  - add Game.look() method  - add static CommandWords.showAll() method and call from Game.printHelp() |
| Add a different command language besides English | Requires changes to:  - Game.processCommand()  - validCommands field in CommandWords | - Game logic tightly tied to English  - Decouple by encapsulating command words as an enumerated type | 1. Convert CommandWords to be enumerated type, adding:  - command field, constructor, 1 enum element for each valid command + unknown  - add methods: getCommand, toString, & static getCommandWord  - update static methods isCommand & showAll  2. Update Command to store a CommandWord field (also update constructor & getCommandWord)  3. Update Parser:  - remove CommandWords field & initialization in constructor  - convert call to isCommand to be a static reference & update call to Command constructor by calling CommandWords.getCommandWord()  4. Update Game.processCommand to use new CommandWord enum instead of Strings  5. Now can update CommandWords to have a number of acceptable command strings, in many languages, by changing the constructor |