User Interface Module Design and Specification

By Kailash Joshi

11/10/2015

Table of Contents

Revision History

Author	Reviewer	Date	Version	Notes
Kailash Joshi	Jamie Alexander	11/10/2015	1.0	Created the document
		25/10/2015	1.1	Added section 2.4

1. Introduction:

This document describes the high level design and interface specification of UI module. It also addresses the assumption and dependencies of the module.

1.1 Module Description: UI is an entry point for the user to DVCS. It provides a set of commands to perform desired action. Finally, it displays the output of that action to the user.

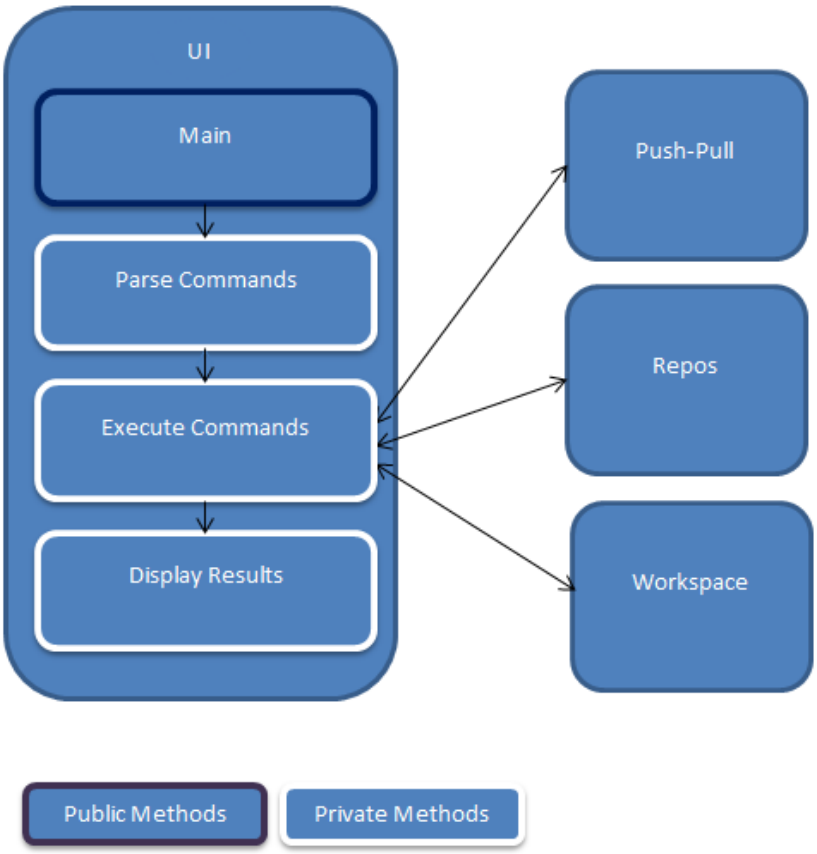
1.2 Responsibilities: UI is responsible for getting commands from users, interpreting the commands, interacting with other modules to get that command executed and finally showing the results back to the user.

1.3 Abbreviations:

- UI – User Interface
- DVCS – Distributed Version Control System

2. Detailed Design:

2.1 Diagram



2.2 Explanation:

UI module is divided into four sub modules:-

- **Main**: - This sub module is responsible for starting the module. It scans for the user input. As soon as an input is found, it is forwarded to the next sub module (Parse Commands).

- **Parse Commands**: - This module gets the user input from the main module in the form of a string. Here the string is parsed and matched for valid commands. If the commands and their options are valid then they are pushed into an array of string and passed to the next module (Execute Commands).

- **Execute Commands**: -This module takes the decision as to which module to call based on the user command. For E.g. if a user inputs a 'Pull' command then this module redirects control to the Push-Pull module for actual execution of the command. In turn, it is also responsible for receiving the response (in form of String) of execution of commands from outer modules. This response is then sent to the next module (Display Result).

- **Display Result**: - It is responsible for showing the output of the command execution to the user.

2.3 Functions to be developed:

- Public **main**() – returns void
- Private **parseCommands**(String input) - returns void
- Private **executeCommands**(String[] tokens) – returns void
- Private **displayResult**(String output) – returns void

2.4 Supported Commands:

Sr. No.	Command	Parameter	Description
1	init		Create an empty Git repository or reinitialize an existing one
2	add		Add files contents to the index
		.	Add all files contents to the index
		<file>…	Add specified file contents to the index
3	checkout		Switch branches or restore working tree files
		-b <branch>	Create a new branch named <branch>
		<branch>	Switch to <branch>
		--track	When creating a new branch, set up "upstream" configuration.
4	commit		Record changes to the repository
		-a	Automatically stage files that have been modified and deleted
		-m <msg>	Use the given <msg> as the commit message
		<file>…	Commit specified <file>

5	branch		List, create, or delete branches
		-a	List both remote-tracking branches and local branches.
		-d	Delete a branch
		<branchname>	The name of the branch to create or delete.
6	merge		Join two or more development histories together
		<commit>…	Commits, usually other branch heads, to merge into our branch
7	push		Update remote refs along with associated objects
		origin <repository>	The "remote" repository that is destination of a push operation
8	pull		Fetch from and integrate with another repository or a local branch
		origin <repository>	The "remote" repository that is the source of a fetch or pull operation
9	status		Show the working tree status
10	clone		Clone a repository into a new directory
		<repository>	The (possibly remote) repository to clone from
		<directory>	The name of a new directory to clone into

3. Dependencies:

- UI depends on **Repos, Push-Pull and Workspace** for the actual execution of the commands. The response from these modules is what is shown to the user by the UI module.
- Sub module **'Execute Command'** will change if the format of the response sent from other module changes.

4. Assumptions:

- Output from **Repos, Push-Pull and Workspace** is received in the form of a string.