This documents the behavior for each of the following dvcs commands/functionalities:

- init: create an empty repository
- clone: copy an existing repository
- add: add specific files that you want to track
- remove: remove specific files from tracking list
- status: check the current status of current repository
- heads: show the current heads
- diff: check the changes between revisions
- cat: inspect a file of a given revision
- checkout: check out a specific revision
- commit: commit changes
- log: view the change log
- merge (challenging): merge two revisions
- pull (challenging): pull the changes from another repository
- push (challenging): push changes into another repository

VALID_COMMANDS :: [String]
The above list contains a hardcoded list of all the valid command names supported by our dvcs system

How each of these commands should like to the user, their syntax and their outcome(response) messages on a terminal should be secrets of the User Interaction module which we document below

Universal format:

```
dvcs $command_name [..$parameters]
```

Each command should begin with the 'dvcs' keyword (lowercase) followed by a command name which can be any one of the above functionality names and an optional list of arguments.
Note that our version of dvcs does not take any flags.

Behavior:

______Here we show the behavior for each possible scenario on the terminal with their output messages.

1. `dvcs`

Entering the above command without any command name/parameters will output a message of how to use dvcs. We check this using the `hasCommand` predicate
Following will be a snapshot of the message, when `hasCommand` returns false:

```
usage: dvcs <command> [<args>]
```

```
These are the dvcs commands used in various situations:

start a working area
   clone        Clone a repository into a new directory
   init         Create an empty DVCS repository

work on the current change
   add          Add files to the tracking set
   remove       Remove files from the tracking set

examine the history and state
   log          Show commit logs
   status       Show the tracked and changed files

grow, mark and tweak your common history
   checkout     Roll back to a commit
   commit       Record changes to the repository
   diff         Show changes between commits
   merge        Join two or more development histories together

collaborate
   pull         Fetch from and integrate with another repository or a
local branch
   push         Update remote repository
```

2. `dvcs $command_name [..$parameters]`

This is the case where `hasCommand` has returned true.
After entering the above command with just a command name will result in searching the command_name in VALID_COMMANDS.

If $command_name not in VALID_COMMANDS, we show the following message:

```
dvcs: '$command_name' is not a dvcs command. See 'dvcs --help’’.
```

If $command_name in VALID_COMMANDS. We have the following cases:

- `dvcs init: () -> Project`
    - Creates repository in CD
    - Failures
        - Repository already exists in CD or anywhere in its ancestor, Message: `Reinitialized existing dvcs repository in <path_name>`
        - Extra arguments after init

    - Success
        - Everything else
- `dvcs clone: String -> Project`
    - Clones repository into CD
    - String has two formats
        - /path/to/repository
        - username@host:/path/to/repository
    - Failures
        - Input/remote directory is not a dvcs repository
        - Cannot connect to host
        - Repository already exists in CD
    - Success
        - Everything else
- `dvcs add: String -> ()`
    - Adds “String=File Name/Path” to TS
    - String can be be a file or directory
    - Failures
        - File or Dir does not exist in CD
    - Success
        - Everything else
- `dvcs remove: String -> ()`
    - Removes “String” from TS
    - String can be a file or directory
    - Failures
        - File or Dir does not exist in TS
    - Success
        - Everything else
- `dvcs status: () -> IO`
    - Prints status of Project to StdOut (terminal)
        - Currently tracked files
        - Non-tracked files
        - Tracked files that have been modified (added, removed, moved)
    - Failures
        - No project in CD
    - Success
        - Everything else
- `dvcs heads: () -> IO`
    - Print’s most recent commit pointed to by HEAD pointer to StdOut.
    - Failures
        - No Repository in CD
        - No commits have been made
    - Success
        - Everything else
- `dvcs diff: () -> IO | String -> IO`

        - If no parameter given
            - Compares CD against Project
        - If “String” is….
            - Unsure
        - Failures
            - No project in CD
        - Success
            - Everything else
- `dvcs log: () -> IO`
    - Success
        - Show the list of commits in a chronological order.
          Each commit will show a commit id, commit message, timestamp, author and set of files as follows:

          ```
          commit <commitId#1>
          Author: cepheid42 <andrew.sexton.42@gmail.com>
          Date:   Sun Nov 17 16:26:22 2019

              My first commit!

          README

          ------------------------------------------------
          commit <commitId#1>
          Author: cepheid42 <andrew.sexton.42@gmail.com>
          Date:   Sun Nov 17 15:50:57 2019

              My second commit!

          README
          Hello.hs

          ------------------------------------------------
          ```
    - Failure
        - Not in a dvcs repository. We show the message:
          `fatal: not a dvcs repository .dvcs`
        - `No commits yet. We show the message:`
          `fatal: You do not have any commits yet`
        - Any argument after log. Error message:
          fatal: ambiguous argument 's' (eg. dvcs log s r)
- `dvcs checkout $commit_id: String -> IO`
    - Success
        - `If $commit_id == ‘’. Do and print nothing`

        - If $commit_id is any valid commit_id, then show the following message:
          Note: checking out <commit_id>.
          HEAD is now at <commit_id> <commit_msg>
        - If $commit_id == ‘leaf’, then show the following message:
          Previous HEAD position was <commit_id> <commit_msg>
          Switched to latest commit.
    - Failure
        - Not in a dvcs repository. Error message:
          fatal: not a dvcs repository .dvcs
        - Not a valid commitId. Error message:
          error: 'er' did not match any commit. (eg. dvcs checkout er/dvcs checkout $commit_id er)
- dvcs commit $commit_message: String -> IO
    - Success
        - Committable Change:
            - Commit created with Id <commit_id>
        - Non committable Change:
            - No tracked file is either new or altered
    - Failure
        - Not in a dvcs repository.
        - No argument after commit
        - Extra argument commit message
        - Nothing to commit even if commit_message is provided
- dvcs cat $commit_id $file_name_path: String -> String -> IO
    - Success
        - Show file contents
    - Failure
        - No filename given, no commit_id given or extra argument given
        - Commid_id or/and file path isn’t valid
- dvcs pull $repo_path: String -> IO
    - Success
        -
    - Failure
        - Not in a dvcs repository.
        - $repo_path is not a dvcs repository
        - $repo_path is not a valid directory path or hostpath
        - No $repo_path given
        - Extra arguments given
        - Local branch is ahead or upto date
        - Any other merge failure
- dvcs push $repo_path: String -> IO
    - Success

    -
- `Failure`
    - Not in a dvcs repository.
    - $repo_path is not a valid directory path or hostpath
    - $repo_path is not a dvcs repository
    - No $repo_path given
    - Extra arguments given
    - Remote branch is ahead or upto date
    - Any other merge failure