There's no difference between list implementations in both of your examples. There's however a difference in a way you can further use variable myList in your code.

When you define your list as:

List myList = new ArrayList();

you can only call methods and reference members that belong to List class. If you define it as:

ArrayList myList = new ArrayList();

you'll be able to invoke ArrayList specific methods and use ArrayList specific members in addition to those inherited from List.

Nevertheless, when you call a method of a List class in the first example, which was overridden in ArrayList, then method from ArrayList will be called not the one in the List.

That's called polymorphism. You can read up on it.

**Most Recent Commit**

The commit ID of the most recent commit is 3884eab839af1e82c44267484cf2945a766081f3. You can use this commit ID to return to the latest commit after checking out an older commit.

**Changing background color**

If you prefer the background color of Git Bash to be something other than black, you can change it in the "Defaults" menu under the "Colors" tab. If you like the background color as-is, you don't need to make any changes.

**Downloading necessary files**

* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-completion.bash) in your home directory with the namegit-completion.bash.
* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-prompt.sh) in your home directory with the name git-prompt.sh.
* Download bash\_profile\_course from the Downloadables section.
* If you already have a file in your home directory named.bash\_profile, copy the content from bash\_profile\_course and paste it at the bottom of .bash\_profile. Otherwise, movebash\_profile\_course to your home directory and rename it to.bash\_profile. (If you're curious to learn more about how bash prompts work, see [this page](http://www.cyberciti.biz/tips/howto-linux-unix-bash-shell-setup-prompt.html).)

**Making Git configurations**

Run the following Git configuration commands. The first one will need to be modified if you are using a text editor other than Sublime, or if Sublime is installed in another location for you. See [this page](https://help.github.com/articles/associating-text-editors-with-git/) for the correct command for a couple of other popular text editors. For any other editor, you'll need to enter the command you use to launch that editor from Git Bash.

git config --global core.editor "'C:/Program Files/Sublime Text 2/sublime\_text.exe' -n -w"

git config --global push.default upstream

git config --global merge.conflictstyle diff3

**Make sure you can start your editor from Git Bash**

If you use Sublime, you can do this by adding the following line to your.bash\_profile:

alias subl="C:/Program\ Files/Sublime\ Text\ 2/sublime\_text.exe"

**Restart Git Bash**

You'll need to close and re-open Git Bash before all your changes take effect.

## Git command review

**Compare two commits, printing each line that is present in one commit but not the other.**

git diff will do this. It takes two arguments - the two commit ids to compare. 

**Make a copy of an entire Git repository, including the history, onto your own computer.**

git clone will do this. It takes one argument - the url of the repository to copy. 

**Temporarily reset all files in a directory to their state at the time of a specific commit.**

git checkout will do this. It takes one argument - the commit ID to restore. 

**Show the commits made in this repository, starting with the most recent.**

git log will do this. It doesn't take any arguments.

If you accidentally add a file to the staging area, you can remove it usinggit reset. For example, if you accidentally add lesson\_2\_reflections.txt, but don’t want it to be committed yet, rungit reset lesson\_2\_reflections.txt and the file will be removed from the staging area, but it will still be in your working directory.