EXERCISE 2

Now that you have created a local git repository, you are ready to create new projects within that repo. Complete the remaining work in the `LearnToCode/Workbook1/Mod01` folder.

Step 1: Create the following directory structure

```
git_exercise/
    ├── pom.xml
    └── src/
        └── main/
            └── java/
        └── test/
            └── java/
```

Step 3: Check the "status" of your repository

Run `git status` and you should see output similar to the following:

```
On branch main

Initial commit

Untracked files:

  (use "git add <file>..." to include in what will be committed)

  pom.xml

nothing added to commit but untracked files present (use "git add" to track)
```

Notice that it shows `pom.xml` as an untracked file. Where are your directories? Git does not keep track of empty directories.

Step 4: Handle empty directories

Add `HelloWorld.java` to the `src/main/java` directory, and a `.gitkeep` file to your `src/test/java` directory. This allows them to be tracked by Git.

Step 5: Check the "status" of your repository using the proper git command.

You should now see your directories being tracked.

```
On branch main

Initial commit

Untracked files:

  (use "git add <file>..." to include in what will be committed)

  pom.xml

  src/

nothing added to commit but untracked files present (use "git add" to track)
```

Step 6: Stage the files so that they can be committed

Run `git add .` to add our files to our staging area to be committed. `git add` is the command to stage files for a commit. `.` is everything in the current directory.

Step 7: Check the "status" of your repository

Run `git status` to see the status of your repository. You no longer have untracked files. You have a list of files to be committed.

```
On branch main

Initial commit

Changes to be committed:

  (use "git rm --cached <file>..." to unstage)

  new file:   pom.xml

  new file:   src/main/java/HelloWorld.java

  new file:   src/test/java/.gitkeep
```

Step 8: Create our first commit

Run `git commit -m "Initial commit"`. This command will take all the files to be committed (staged files) and create a point in history related to these changes.

When you create a commit, it should encompass all the changes related to a certain task or logical set of changes

Step 9: Check the log

If you run `git log`, you will see your commit history

```
commit bddf8d41d0f8f6d0c9ebdb0677e2913b55da9311
Author: Dana Wyatt <dana.wyatt@myemailprovider.com>
Date:   Wed Feb 15 16:36:20 2017 -0500
    Initial commit
```

Step 10: Edit `HelloWorld.java`

Open the project in IntelliJ.

Then, open `HelloWorld.java` and add this basic structure to the file

```
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World!");
    }
}
```

Step 11: Check the "status" of your repository

Run `git status` to see that you have modified the `HelloWorld.java` file since your last commit

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
  modified:   HelloWorld.java
no changes added to commit (use "git add" and/or "git commit -a")
```

Step 12: View the differences between now and your last commit

Run `git diff` to see what has changed in the file(s) that have been modified. Lines added to the file will have a **+** next to them. Lines removed from the file will have a **-** next to them.

Step 13: Stage the new changes for commit and review repo status

Let's stage these changes using `git add .`. Run `git status` to now see that the files are listed as changes to be committed.

```
On branch main
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
  modified:   index.html
```

Step 14: Commit the changes to `HelloWorld.java`

Commit your changes with a meaningful message.

```
git commit -m "added basic java code to HelloWorld.java"
```

Step 15: Check the commit log

Run `git log` and see that we have another entry in the log for our new commit

```
commit 789b028b04bfe0efc080827855dd00f79d20433d
Author: Dana Wyatt <dana.wyatt@myemailprovider.com>
Date:   Wed Feb 15 16:47:32 2017 -0500

    added basic html to index.html


commit bddf8d41d0f8f6d0c9ebdb0677e2913b55da9311
Author: Dana Wyatt <dana.wyatt@myemailprovider.com>
Date:   Wed Feb 15 16:36:20 2017 -0500

    Initial commit
```

Step 17: Compare commits with 'git diff'

Run `git diff` to see what was different between commits. You will need to provide the actual hash from the commits you are comparing. You can find them in the commit log.

```
$ git diff INITIAL_COMMIT_HASH SECOND_COMMIT_HASH
```