SOEN 345 - Software Testing, Verification, and Quality Assurance
Assignment 6

Ali Hanni - 40157164

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Winter 2022

*** NOTE ***

- All requested images are joined with this submission under the folder `/img/`
- The email received after an unsuccessful build on Jenkins is also joined as: `partE_mailBuildFail.pdf`
- My github account is alimus22 (https://github.com/alimus22)
- The email address used for this assignment is: alihanni95@gmail.com

Section 1

Question 1

Even though Maven is very efficient at building projects and executing tests, Jenkins is all around better and covers all aspects of CI/CD. In my opinion, it is easier and seamless to install the plugins you want and view the results with Jenkins. The different options easily accessible (like receive email when build fails) also makes a difference. We can set our environment (JDK, Maven etc) easily, which can help avoid many mistakes. In fact, Jenkins uses Maven as a tool and incorporates (and expands on) most of its functionalities if not all.

Question 2

First, having a history of previous build makes it possible to quickly see the differences between each build, for examples differences in test coverage which can help to more easily understand the cause of fluctuations. This in fact can be applied to a wide variety of other development activities, not just testing. In case of failure, we can compare previous failing builds with current in order to look for similarities.

Question 3

We could of course use a platform like Jenkins to verify and execute integration tests. For example, after packaging the code, Jenkins can upload it to other (internal) repos with other pieces of software. Jenkins supports a multitude of different environments (different operating systems for example) and run the integration tests. Jenkins could also be used to automate repetitive tasks as backing up a database with interesting plugins that let you manage your backups.

Question 4

The Docker Pipeline plugin seems very interesting. In many projects, I use Docker in order to have the exact same environment across multiple systems. It is particularly helpful while building machine learning projects with python that requires dozens of different python libraries to work. It is easier to have everything within one container and simply load the image when changing machine. Even though python is not a compiled language and therefore projects in python do not need to be built, Jenkins provide a lot of interesting functionalities that can still benefit any project and the possibility to have a plugin that offers a domain-specific-language to mimic Docker's most used operations seems very handy.

Section 2

Question 1

It took in total 4 iterations to find the faulty commit. Note that we could have found the faulty commit in 3 iterations but the extra iterations allowed us to verify that the build for that particular commit fails, and to receive information about the faulty commit.
The complete process of bisecting is joined with this submission as `bisecting.txt`. Note that comments on if the particular build failed or succeeded are added.

Question 2

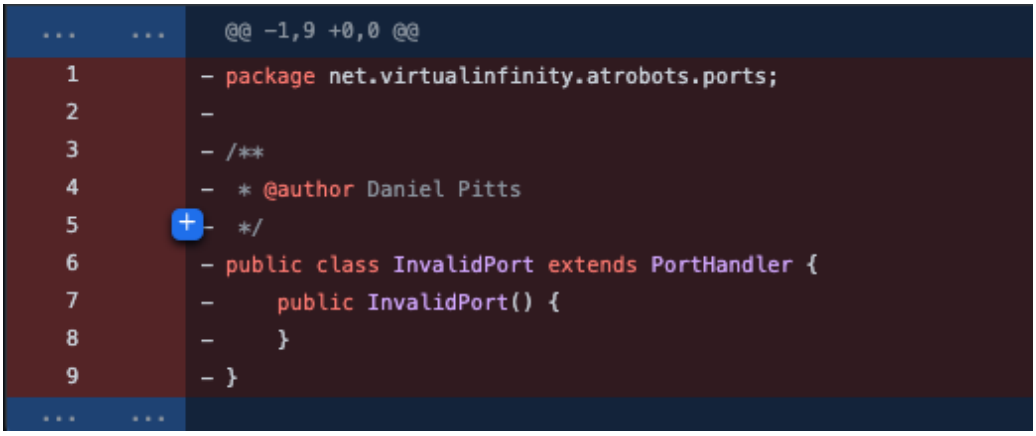
First after `git bisect start` we provide a 'good' commit (i.e. a3bbceff0a9ea73215a4d 6ff54f29e0aee102e27) and the command `git bisect bad` sets the current commit as bad. Git does the binary search and checkout to a commit in between. We need to rebuild (`ant clean compile`) in order to verify if that particular commit is 'good' or 'bad' and we specify the result to git bisect using the appropriate command in order for the process to continue. We do that by using `git bisect good` or `git bisect bad` that sets current commit as respectively good or bad. After a few iterations, the binary search is done and we are supposed to be on the faulty commit.
In this case, the faulty commit seems to be: 522a07e4dce45efe9ae29b740c763069c3d6d047, named 'Inline class refactoring'.

Question 3

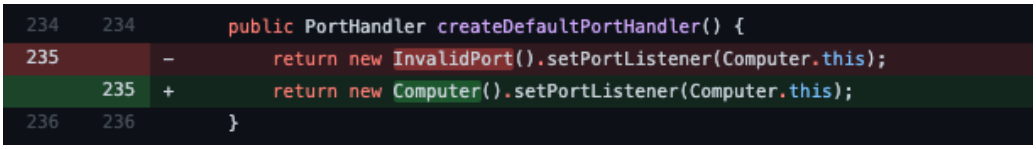
```
alihanni@Ali at-robots % git bisect bad
522a07e4dce45efe9ae29b740c763069c3d6d047 is the first bad commit
commit 522a07e4dce45efe9ae29b740c763069c3d6d047
Author: unknown <rodrigomorales2@acm.org>
Date:   Wed Mar 24 18:05:58 2021 -0400

    Inline class refactoring

 .../java/net/virtualinfinity/atrobots/computer/Computer.java     | 5 ++++-
 .../java/net/virtualinfinity/atrobots/ports/InvalidPort.java     | 9 ---------
 2 files changed, 4 insertions(+), 10 deletions(-)
 delete mode 100644 src/main/java/net/virtualinfinity/atrobots/ports/InvalidPort.java
alihanni@Ali at-robots % git bisect reset
Previous HEAD position was 522a07e Inline class refactoring
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
alihanni@Ali at-robots % _
```

Question 4

In this commit, the `InvalidPort` class is deleted.



A call to create an instance of `InvalidPort` is replaced by one to create an instance of `Computer`



However, since `Computer` is not well defined and do not extend `PortHandler`, the call `Computer().setPortListener(Computer.this)` is not possible since the method `setPortListener()` is defined in `PortHandler`. `Computer` implementd `PortListener` that do not have such method which will cause the error. In order to fix it, `Computer` must extend `PortHandler` which is possible since `Computer` do not extend any other class.