Name: Sanaya Vakharia **PID:** A17412057 **(page 1)**

NumberServer.java
(Server.java not shown)

done in lab

```
import java.io.IOException;
import java.net.URI;

class Handler implements URLHandler {
  int num = 0;
  public String handleRequest(URI url) { /* handles requests for /, /increment, /add?count=X */ }
}

class NumberServer {
  public static void main(String[] args) throws IOException { /* starts up the server on a port given in args[0] }
}
```

TestNumberServer.java

```
import static org.junit.Assert.*;
import org.junit.*;
import java.net.URI;
import java.net.URISyntaxException;

public class TestNumberServer {
  @Test
  public void testIncrement() throws URISyntaxException {
    Handler h = new Handler();
    URI increment = new URI("http://localhost/increment");
    URI rootPath = new URI("http://localhost/");
    assertEquals("Number incremented!", h.handleRequest(increment));
    assertEquals("Number: 1", h.handleRequest(rootPath));

    assertEquals(__________“Number is incremented!”__________, h.handleRequest(increment));

    assertEquals(__________“Number: 2”__________, h.handleRequest(rootPath));
  }
}
```

Text

test.sh

```
set -e
javac -cp .:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar Server.java NumberServer.java TestNumberServer.java
java -cp .:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar org.junit.runner.JUnitCore TestNumberServer
```

start.sh

```
set -e
javac Server.java NumberServer.java
java NumberServer 4001
```

```
$ bash test.sh
JUnit version 4.13.2
...
Time: 0.007

OK (3 tests)
```

```
$ bash start.sh
Server Started! Visit http://localhost:4001.
```

Terminal Interaction

What do you notice and wonder about this program and these commands? What problems do they solve?

At its simplest, a **bash script** (or **shell script**) is a sequence of commands we could run at the terminal saved in a file, usually with .sh extension.

We can run them all by using bash from the terminal on that file. It can save us a lot of typing and remembering commands. We can save bash scripts in repositories to make it easy to build after cloning.

Goals/Outcomes:
- type bash test.sh instead of java-cp and java
- save the commands needed to run + test the proj

Name: Sanaya Vakharia **PID:** A17412057 **(page 2)**

What does `set -e` do and why should we care?

changed bash's behavior to stop after the first command with error. avoids confusing mix of success / failure output

```
$ # remove set -e, edit NumberServer.java to have a missing ";"
$ bash test.sh
NumberServer.java:11: error: ';' expected
            return String.format("Number: %d", num)
                                                   ^
1 error                          error
JUnit version 4.13.2
...
Time: 0.007
                     test pass
OK (3 tests)
```

What if we want to provide the port? How should we change start.sh below to accomplish that?

```
$ bash start.sh 8765
Server Started! Visit http://localhost:8765 to visit.
```

start.sh

```
set -e

javac Server.java NumberServer.java

java NumberServer 4001 $1
```

This has a long list of `.java` files – what if we add another one? Any way to type less?

test.sh

```
set -e

javac -cp .:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar Server.java NumberServer.java TestNumberServer.java


java -cp .:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar org.junit.runner.JUnitCore TestNumberServer
```