NumberServer.java
(Server.java not shown)

```
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 incremented!"_____, h.handleRequest(increment));

    assertEquals(__"Number: 2"_____, h.handleRequest(rootPath));
  }
}
```

test.sh

```
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

```
javac Server.java NumberServer.java
java NumberServer 4001
```

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

OK (1 test)
```

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

echo .sh --> prints its arguments (after resolving them)

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

what is ".sh" and what does it stand for? if "start.sh" is in a directory (if), do we have to be in that directory?

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.

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.
```

$# --> number of arguments

start.sh

```
set -e  --> exits the bash script if there is a bash error
          should put at the top of .sh file
javac Server.java NumberServer.java

java NumberServer __$1--> means the first command line argument in bash
```

bash would keep running even if there is error

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

# instead write...
```

"pattern" --> *.[file extension]

```


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

echo *.java -->expands of all files