Name: **PID:** **(page 1)**

```
1   // imports...                                          GradeServer.java
2     /**
3          Takes a command, represented as an array of strings
           runs it, and returns its combined stdout and stderr as a
4          string.
5     */
6     static String exec(String[] cmd) throws IOException {
7          Process p = new ProcessBuilder()
8
9
10                        .command(Arrays.asList(cmd))
11
12
13                        .redirectErrorStream(true)
14
15
16                        .start();
17         InputStream outputOfBash = p.getInputStream();
18         return new String(outputOfBash.readAllBytes());
19    }
20
21  }
22  class ExecExamples {
23    public static void main(String[] args) throws IOException {
24      String[] cmd1 = {"ls", "lib"};
25      System.out.println(ExecHelpers.exec(cmd1));
26
27      String[] cmd2 = {"pwd"};
28      System.out.println(ExecHelpers.exec(cmd2));
29
30      String[] cmd3 = {"echo hello world”};
31      System.out.println(ExecHelpers.exec(cmd3));
32
33      String[] cmd4 = {"touch", "a-new-file.txt"};
34      System.out.println(ExecHelpers.exec(cmd4));
35    }
36  }
```

directory structure

```
list-examples-grader/
  lib/
    hamcrest-core-1.3.jar
    junit-4.13.2.jar
  GradeServer.java
  Server.java
  TestListExamples.java
  grade.sh
```

Which line creates and starts a **`Process`**?

Line 16.

```
$ javac GradeServer.java Server.java
$ java ExecExamples
hamcrest-core-1.3.jar
junit-4.13.2.jar


/Users/joe/src/list-examples-grader

hello world
```

What other interesting effects would happen in addition to this output?

a-new-file.txt, empty, would be created in the directory.

`public ProcessBuilder command(List<String> command)` — *In ProcessBuilder class*

Sets this process builder's operating system program and arguments.

`public ProcessBuilder redirectErrorStream(boolean redirectErrorStream)` — *In ProcessBuilder class*

Sets this process builder's redirectErrorStream property. If this property is true, then any error output generated by subprocesses subsequently started by this object's start() method will be merged with the standard output, so that both can be read using the Process.getInputStream() method.

`public Process start() throws IOException` — *In ProcessBuilder class*

Starts a new process using the attributes of this process builder.

`public abstract InputStream getInputStream()` — *In Process class*

Returns the input stream connected to the normal output of the subprocess. The stream obtains data piped from the standard output of the process represented by this Process object.

GradeServer.java

```
list-examples-grader/
  lib/
    hamcrest-core-1.3.jar
    junit-4.13.2.jar
  GradeServer.java
  Server.java
  TestListExamples.java
  grade.sh
```

directory structure

```
1   // imports...
2   class ExecHelpers {
3     static String exec(String[] cmd) throws IOException {
4       Process p = new ProcessBuilder()
5                       .command(Arrays.asList(cmd))
6                       .redirectErrorStream(true)
7                       .start();
8       InputStream outputOfBash = p.getInputStream();
9       return String.format("%s\n", streamToString(outputOfBash));
10    }
11    static String streamToString(InputStream out) { /* ... */  }
12  }
13  class Handler implements URLHandler { /* hidden */ }
14  class GradeServer { /* hidden */ }
15  class ExecExamples {
16    public static void main(String[] args) throws IOException {
17      String[] cmd1 = {"ls", "lib"};
18      System.out.println(ExecHelpers.exec(cmd1));
19      // hid the rest of these examples
20    }
21  }
22  // What if we wanted to use Java, rather than grade.sh, to do the grading?
23  // Use the grade.sh at the bottom of this page for reference
24  // For example, ExecHelpers.exec({"rm", "-rf", "student-submission"}); could be used to
25  // run the first `rm` command in that script
26  class Grade {
27    static String grade(String repo) throws IOException {
28      String CPATH = ".:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar";
29      String result = "";
30      result += ExecHelpers.exec({"rm", "-rf", "student-submission"});
31
32
33
34
35
36        result += ExecHelpers.exec({“git”, “clone”, repo})
37
38        result += “Finished cloning\n”;
39
40        result += ExecHelpers.exec({ “cp”, “student-submission/LIstExamples.java”, “./”});
41        result += ExecHelpers.exec({ “javac”, “-cp”, CPATH, “*.java”});
42        result += ExecHelpers.exec({ “java”, “-cp”, CPATH, “org.junit.runner.JUnitCore”, “TestListExamples” });
43
44        return result;
45
46
47
48
49
50
51
52
53    }
54    public static void main(String[] args) throws IOException {
55         System.out.println(grade(args[0]));
56    }
```

One goal is to be able to run:

```
$ java Grade https://github.com/...

Cloning into 'student-submission'...

Finished cloning
```

```
CPATH='.:lib/hamcrest-core-1.3.jar:lib/junit-4.13.2.jar'

rm -rf student-submission
git clone $1 student-submission
echo 'Finished cloning'

cp student-submission/ListExamples.java ./
javac -cp $CPATH *.java
java -cp $CPATH org.junit.runner.JUnitCore TestListExamples
```