Name: Kevin Tran | **PID:** A16809193 | **(page 1)**

Processes actually have two different default places to print: standard **output** and standard **error**

Programming languages typically have a way to select one or the other when printing. In Java, there's `System.err.print`

Many programs, when they report an error message, print to standard **error** (also called **stderr**).

We can observe this difference with redirection:

`cmd 2>file.txt` redirects the output of the command's **stderr** to `file.txt`

`cmd >file.txt 2>&1` redirects **stderr** to **stdout** and then both to `file.txt`

```
class PrintThenThrow {
  public static void main(String[] args) {
    System.out.println("Hello!");
    System.err.println("Error!");
    throw new RuntimeException("This is an error!");
  }
}
```

PrintThenThrow.java

```
$ java PrintThenThrow >out.txt
Error!
Exception in thread "main" java.lang.RuntimeException:
This is an error!
    at PrintThenThrow.main(PrintThenThrow.java:4)
$ cat out.txt
```

Hello!

```
$ java PrintThenThrow >out.txt 2>&1
$ cat out.txt
```

PrintThenThrow: stderr --> stout --> out.txt

```
Hello!
Error!
```

>& --> to redirect a stream to another file descriptor

```
Exception in thread "main" java.lang.RuntimeException:
This is an error!
    at PrintThenThrow.main(PrintThenThrow.java:4)
```

```
class CompileError {
  public static void main(String[] args) {
    int x = "not-a-number";
  }
}
```

CompileError.java

```
$ javac CompileError.java >error-output.txt
CompileError.java:3: error: incompatible types:
String cannot be converted to int
       int x = "not-a-number";
               ^
1 error
$ cat error-output.txt # nothing in this file!
$ javac CompileError.java > error-output.txt
$ cat error-output.txt
CompileError.java:3: error: incompatible types:
String cannot be converted to int
       int x = "not-a-number";
               ^
1 error
```

0 - stdin
1 - stdout
2 - sterr

what if you want to separate stderr and stdout?
java PrintThenThrow > justout.txt 2> justerror,txt

Brainstorm: What makes a good autograder script? How might it work?

Hint – imagine this setup. Gradescope runs:

`$ bash grade.sh <student-github-url>`

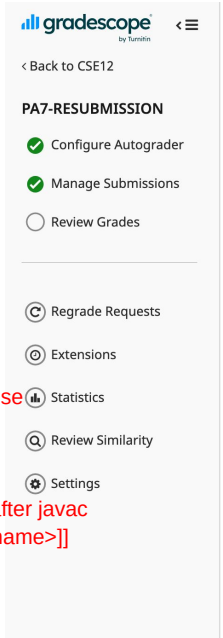
and whatever the text output of that command is gets sent back to the student. What are the steps to, say, grade a PA.

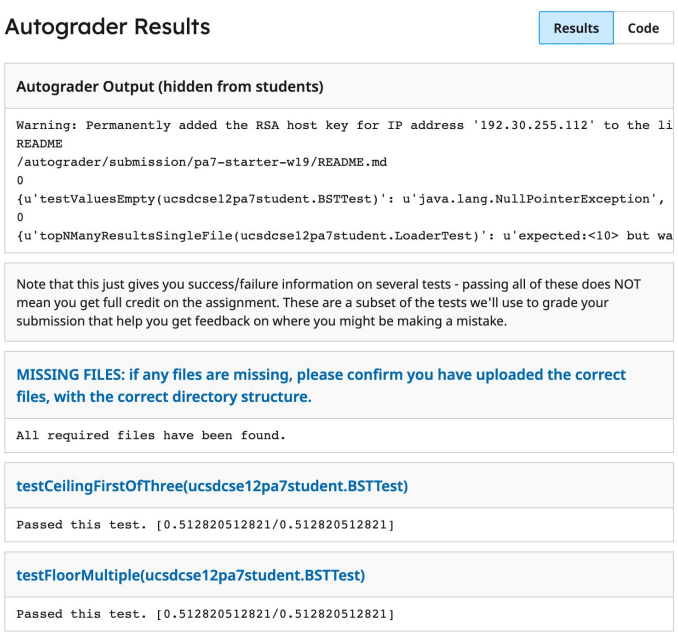
Imagine students submitted their DocSearchServer that we used in lab that searches files, for example. What should the grader do?

- Run a bunch of cases to compare expected + actual
- check compiles correctly, includes all expected files
- Report specific failures
- make sure clone/download worked
- calculate a grade
- somehow report/send grade to gradescope

JUnit or use .expect

check $? after javac
if [[-f <filename>]]





Perform JUnit like tests in BASH. To do this we will first clone from the agecalc repository on GitHub. Let's clone the repo by running the following commands
$ git clone https://github.com/ucsd-cse15l-f23/agecalc.git
$ cd agecalc
Let's now explore our java program's behavior

Assume we have a program with the following behavior. Underlined text is typed at standard input

```
$ javac AgeCalc.java
$ java AgeCalc
1987/6/22
You're 36 yrs old.
$ java AgeCalc
2024/7/12
You don't exist yet.
```

Assume we have a directory of files like below, where the contents of each file is in quotes next to it. **~~Which expectation below is incorrect?~~**

```
AgeCalc.java
check.sh
test-files/
  test1.txt
  test1.txt.expect
  test2.txt
  test2.txt.expect
  test3.txt
  test3.txt.expect
  ...
```

Write a **bash script** that will run the program on all the test files. Print nothing for passing tests, "Test Failed!" with the mismatched output if it doesn't match the expectation, and "Test errored" with the error output if the program had an error.

```
set -e

javac AgeCalc.java
for _____________________________




done
```

Final challenge: Extend the above bash script to count the number of passed and failed tests, and print an overall message about the "grade" of the student's AgeCalc. Hint: We can perform arithmetic in BASH by using $ and double parenthesis.
$ PASSED=0
$ PASSED=$(($PASSED+1));