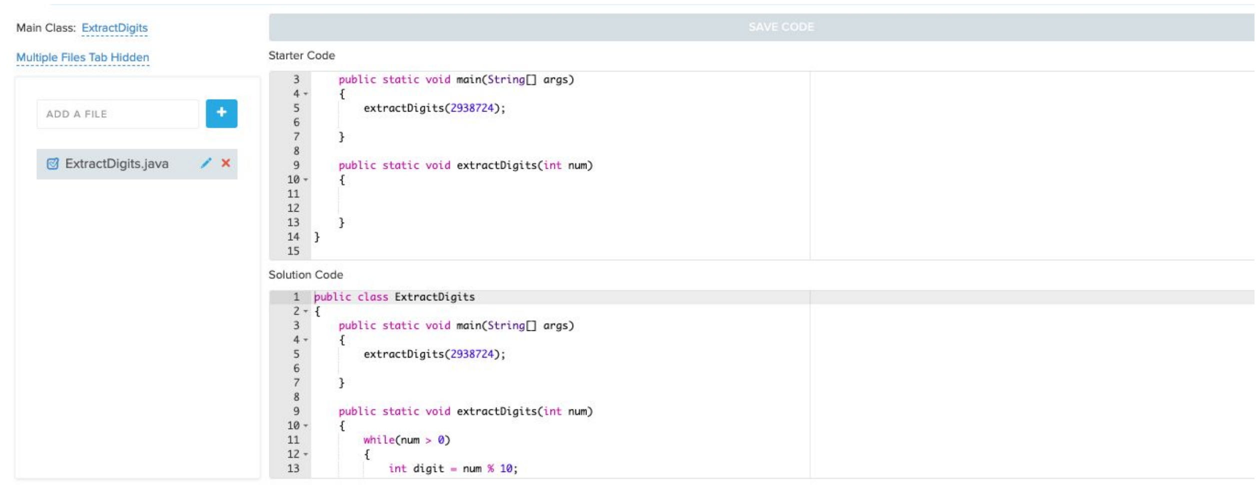
**Basics**:
Autograders in Java rely on reading methods and output from the different class files that students use to complete each exercise. In general, in order to create a Java autograder, exercises need to have the solution code already written.

There are multiple ways to write autograders, but the simplest example of writing one relies on students completing a method (function) as instructed by the exercise.



Starter Code

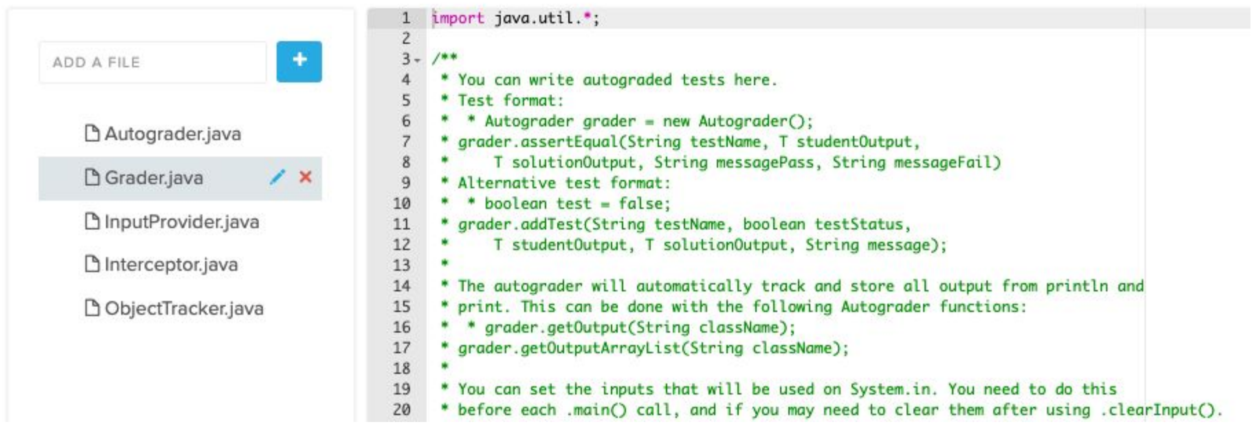
```
    public static void main(String[] args)
    {
        extractDigits(2938724);

    }

    public static void extractDigits(int num)
    {


    }
}
```

Solution Code

```
public class ExtractDigits
{
    public static void main(String[] args)
    {
        extractDigits(2938724);

    }

    public static void extractDigits(int num)
    {
        while(num > 0)
        {
            int digit = num % 10;
```

In the example above, students are expected to write a method, extractDigits, in order to complete this exercise. The method is then tested in the main method in the *same* class file.

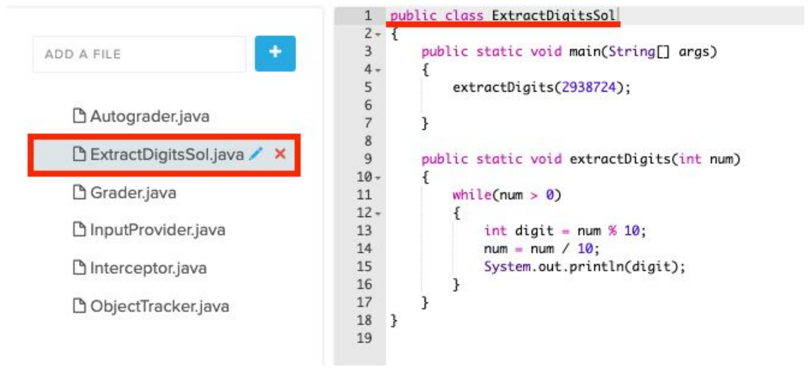
To create an autograder for this problem, we need to ADD AUTOGRADER click in the autograder tab, which will bring up this page:



```
import java.util.*;

/**
 * You can write autograded tests here.
 * Test format:
 *  * Autograder grader = new Autograder();
 * grader.assertEqual(String testName, T studentOutput,
 *     T solutionOutput, String messagePass, String messageFail)
 * Alternative test format:
 *  * boolean test = false;
 * grader.addTest(String testName, boolean testStatus,
 *     T studentOutput, T solutionOutput, String message);
 *
 * The autograder will automatically track and store all output from println and
 * print. This can be done with the following Autograder functions:
 *  * grader.getOutput(String className);
 * grader.getOutputArrayList(String className);
 *
 * You can set the inputs that will be used on System.in. You need to do this
 * before each .main() call, and if you may need to clear them after using .clearInput().
```

The Grader.java file is the file that you will use to write autograders. The Autograder.java file contains useful methods that you can use to help write autograders, and the other files are used to help gather input information and objects from the code editor. The only file that you will write in is the *Grader.java* file.

Before writing an autograder, you will need to create a new file that will host the exercise solution:



```java
public class ExtractDigitsSol
{
    public static void main(String[] args)
    {
        extractDigits(2938724);

    }

    public static void extractDigits(int num)
    {
        while(num > 0)
        {
            int digit = num % 10;
            num = num / 10;
            System.out.println(digit);
        }
    }
}
```

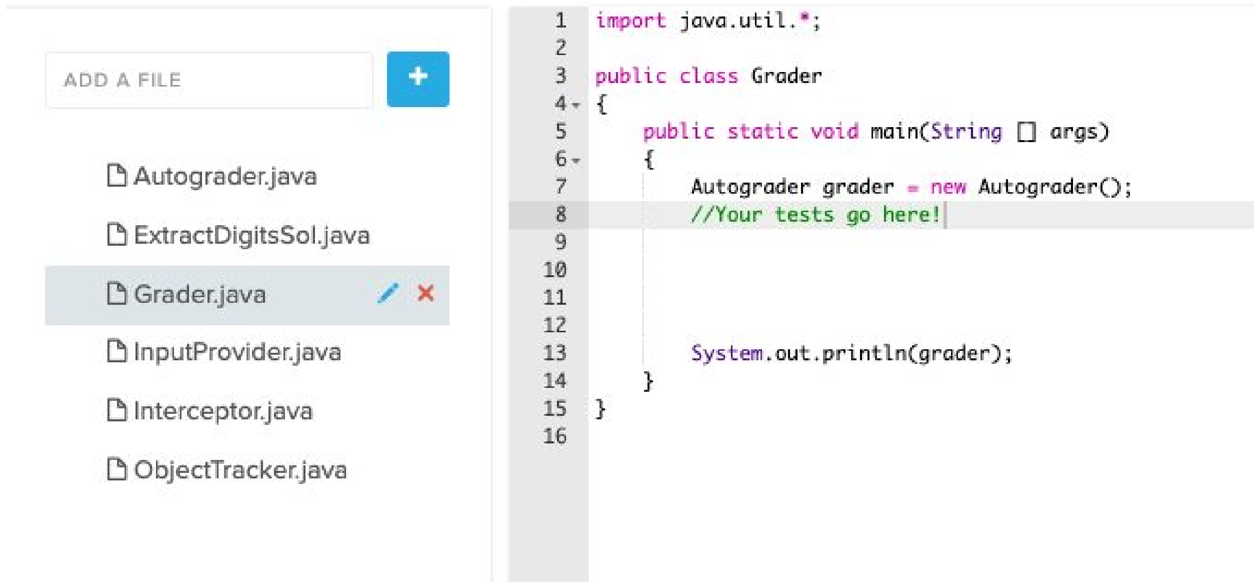
It's useful to name the file similarly to the name in the student code. For example, the file name for students is ExtractDigits.java, so the solution should be ExtractDigitsSol.java.

Now that we have the solution code written, we can write an autograder to test to see if the student has completed the assignment correctly. In this case, we are going to want to check two things:

1. Does the extractDigits method have the correct output based on the description?
2. Does the main method print the correct output?

To test these two things, we are going to test if the student's extractDigits method produces the same output as the solution method, and if the student's main method produces the same output as the solution output.

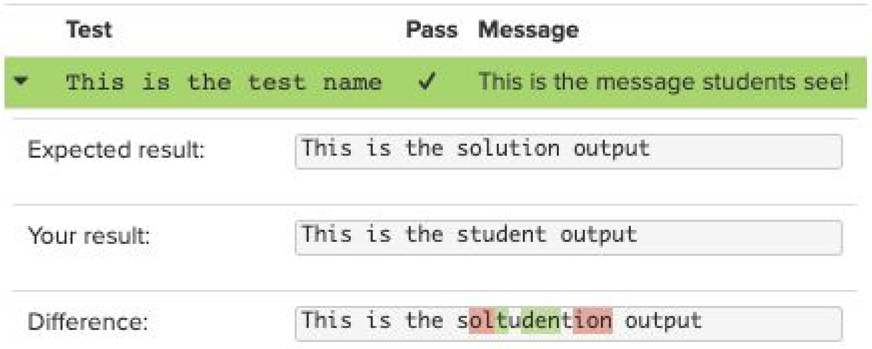
To test that, we will need to write autograder tests in the Grader.java file:



```java
import java.util.*;

public class Grader
{
    public static void main(String [] args)
    {
        Autograder grader = new Autograder();
        //Your tests go here!




        System.out.println(grader);
    }
}
```

There are two ways to write an autograder test - grader.addTest, and grader.assertEqual. The addTest method takes a boolean value and presents the student with a pass/fail depending on the value of the boolean. assertEqual compares two values and determines if they are equivalent:

```
grader.addTest("This is the test name", true, "This is the student output",
"This is the solution output", "This is the message students see!");
```



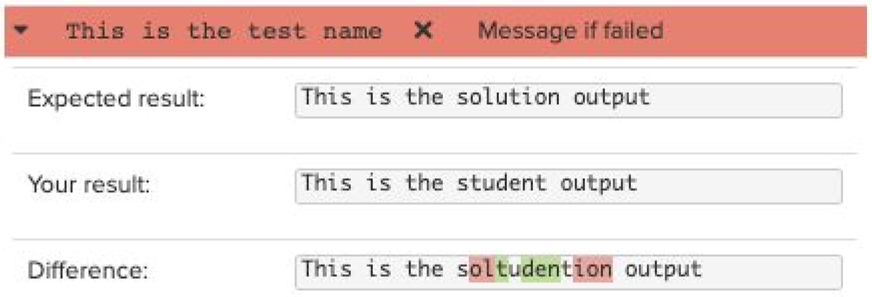
| Test | Pass | Message |
|---|---|---|
| This is the test name | ✓ | This is the message students see! |

Expected result: This is the solution output

Your result: This is the student output

Difference: This is the soltudention output

```
grader.assertEqual("This is the test name", "This is the student output",
"This is the solution output", "Message if passed!", "Message if failed");
```



This is the test name ✕ Message if failed

Expected result: This is the solution output

Your result: This is the student output

Difference: This is the soltudention output

To get student output from student programs, we can create an object using the student class:

```
ExtractDigits student = new ExtractDigits();
```

Once we do that, we can then use the methods associated with the student class.

```
student.extractDigits(12345);
```

and test that versus the solution output:

```
ExtractDigitsSol solution = new ExtractDigitsSol();
solution.extractDigits(12345);
```

In this particular case, the output from the student and solution code is printed in the console. In order to access output to the console, we need to use the code grader.getOutput.

grader.getOutput takes the **class name** as its parameter, and returns a String of all the outputs for a given class.

For this autograder, we could write:

```
String studentOutput = grader.getOutput("ExtractDigits");
```

```
String solutionOutput = grader.getOutput("ExtractDigitsSol");
```

It's important to note that this needs to be called *after* the methods that print to the console is executed. If you want to run multiple tests to test output, the call `grader.clearOutput();` needs to be made so that the console removes all previous output. This should be called *before* the new print calls to the console.

Now that we have successfully gotten the student and solution output from the grader, we can run a test to see if the values are equivalent:

```
grader.assertEqual("Extract Digit has the correct output?", studentOutput,
solutionOutput, "Great!", "Try again!");
```