Mastery CourseGrades Reflection Log

Credit Name: CSE 2120 Data Structures 1
Assignment Name: Mastery CourseGrades

How has your program changed from planning to coding to now? Please explain?

From planning to coding I took the definition of the application from the computer science textbook and started to create each part of code to meet the necessary requirements. Each part became more complex as I made a line of code for the two dimensional array. Although I originally couldn't access the average grade of the 12th student and the 5th test I fixed this after considering that the grades array is 0 based (explained more in my error log)

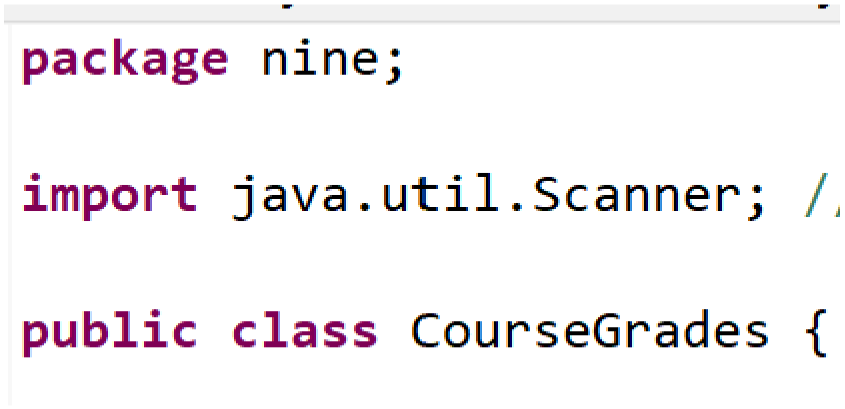
From coding to now I did not change any of the code besides the error I encountered regarding printing the 12th students average and the 5th test average, I changed this code to make the program not output an error (explained more in my error log)

Steps

CourseGrades

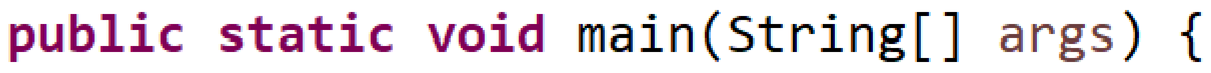
(the package this file is in is Mastery)(NOTE: since I did this at home and my eclipse isn't linked my package is called nine but in the code I submit it will be Mastery)

I started off by importing java.util.scanner which allows me to get different data types from user inputs. The public class is declared as CourseGrades (It's declared as public so the GradeBook class can access its information and return information).



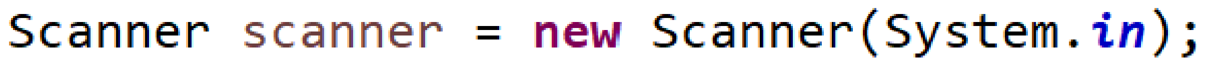
```
package nine;

import java.util.Scanner; /

public class CourseGrades {
```

Below it shows the public (accessible from other classes) method "main" is static (you can access it without creating an object), void (method doesn't return any value), main (method name), and String[ ] args (String represents the data type [ ] indicates that it will be an array, and args (arguments) is the name of the array).



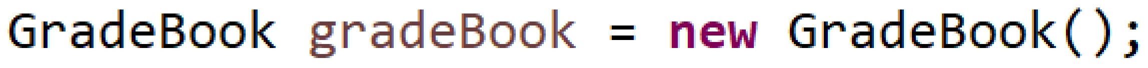
```java
public static void main(String[] args) {
```

The Scanner object is created below to take in inputs from the user. (I named mine scanner)



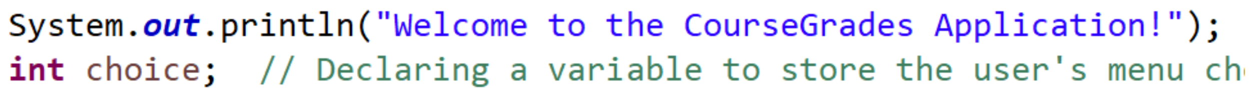
```java
Scanner scanner = new Scanner(System.in);
```

I then created the gradeBook object.
-GradeBook is the type of object created (is the GradeBook class which contains the methods and variables for calculations).
-gradeBook is the name of the object that the GradeBook class above is assigned to (GradeBook classes(and now objects) values will become gradeBook objects values as well)
-new is used to create memory(space) to store the gradeBook object
-GradeBook() this is a constructor that initializes the new object (filling the memory(space) with information for the object from the class GradeBook)



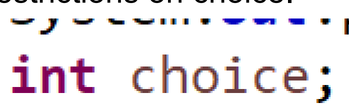
```java
GradeBook gradeBook = new GradeBook();
```

After this I created a welcome statement for the user, this is outside the loop because it doesn't make sense to constantly welcome the user. Doing outside ensures this only prints at the start.



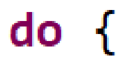
```java
System.out.println("Welcome to the CourseGrades Application!");
int choice;  // Declaring a variable to store the user's menu ch
```

Next, I declared the integer variable choice. I do this outside of the do-while loop, because you need to keep track of the user's input across multiple iterations(loops). If you did it inside of the loop you would lose the previous value every time the loop ran again. Also the while part of the loop needs choice outside of the loop to check its restrictions on choice.



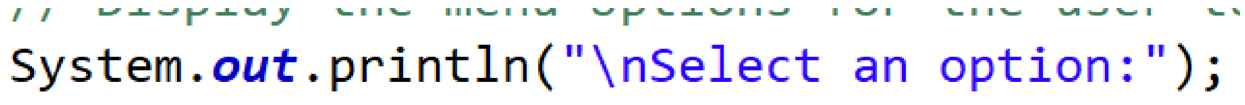
```java
int choice;
```

I then began the do-while loop after declaring and printing the two previous things (welcome message and int choice)



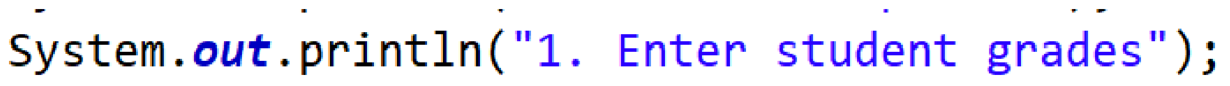
```
do {
```

Next, I made a print statement that prompted the user to select one of the options that will follow.



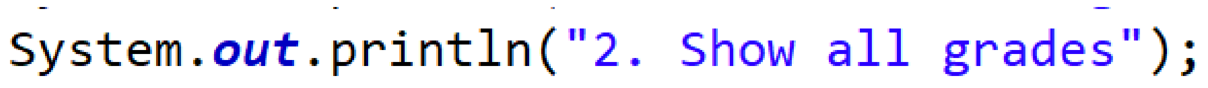
```
System.out.println("\nSelect an option:");
```

After this I created the first option (1), which would allow the user to enter the 5 test grades for one of the 12 students.



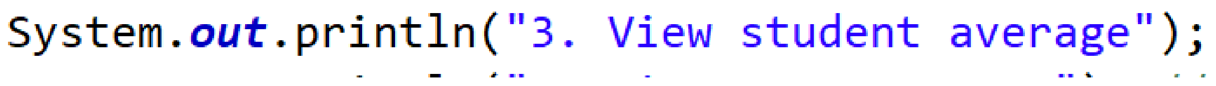
```
System.out.println("1. Enter student grades");
```

Next, I created the second option (2), which would allow the user to see all 12 of the students and the grades the got on the 5 tests in (information will be displayed in a chart/table format)



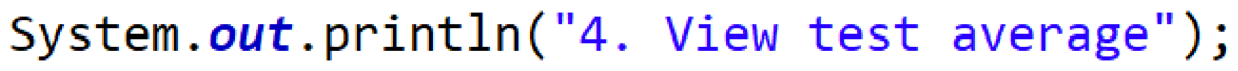
```
System.out.println("2. Show all grades");
```

I then created the third option (3), which would allow the user to view the average grades of a student they select.



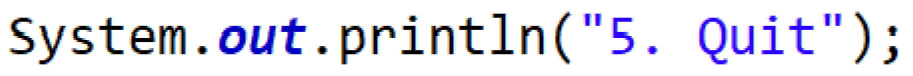
```
System.out.println("3. View student average");
```

After this I created the fourth option (4), which would allow the user to view the average grades of a test they select.



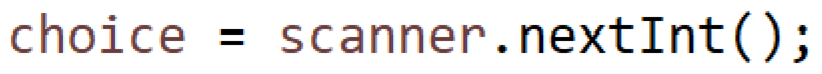
```
System.out.println("4. View test average");
```

Lastly, I made the option to quit (5) (stop the program) for when the user wants to stop using/adjusting things in the grades.



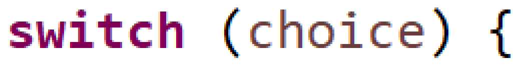
```
System.out.println("5. Quit");
```

I then saved their response, and made this value the value of the integer variable choice that was declared outside of the do-while loop. Note: the name of the Scanner object has to match here hence why I put scanner.nextInt().



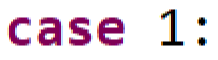
```
choice = scanner.nextInt();
```

I then began a switch statement which used the parameter choice to choose its case (Ex. if the user selected choice 1 then the case that would be executed is 1)



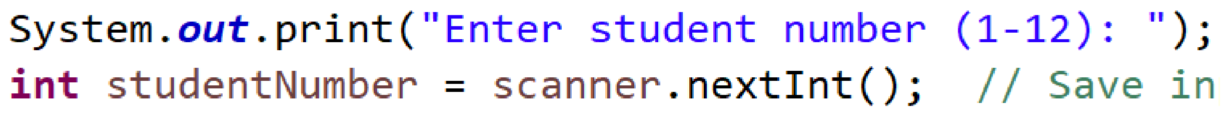
```
switch (choice) {
```

I then created case 1.



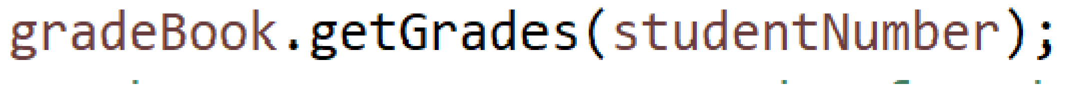
```
case 1:
```

After this, I prompted the user to enter the students number that they wanted to enter the test grades for and saved there response as an integer variable called studentNumber.



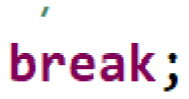
```
System.out.print("Enter student number (1-12): ");
int studentNumber = scanner.nextInt();  // Save in
```

I then called on the method getGrades() which was passed the parameter studentNumber from the gradebook object. (the user will be able to input the test grades from here on which is explained in the GradeBook class reflection)



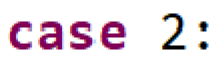
```
gradeBook.getGrades(studentNumber);
```

Lastly, I ended the case with a break line.



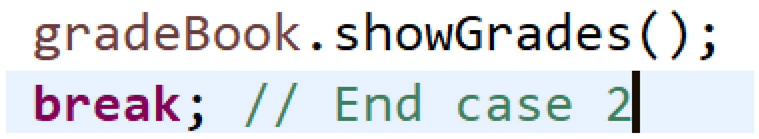
```
break;
```

After this I created case 2.



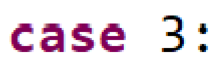
```
case 2:
```

If showing the class grades was selected, then I would call on the showGrades() method from the gradeBook object which would display the grades and students in a chart format (explained more in the GradeBook reflection log). I followed this with a break line to end the case.



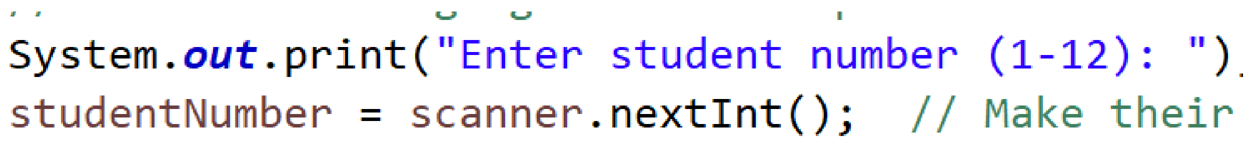
```
gradeBook.showGrades();
break; // End case 2
```

Next, I created case 3.



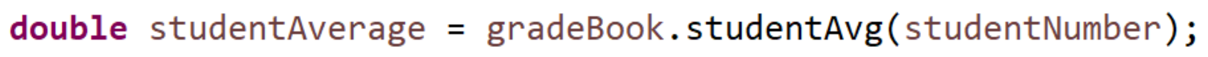
```
case 3:
```

I then prompted the user to enter which student they wanted to know the average of and saved this as the variable studentNumber.



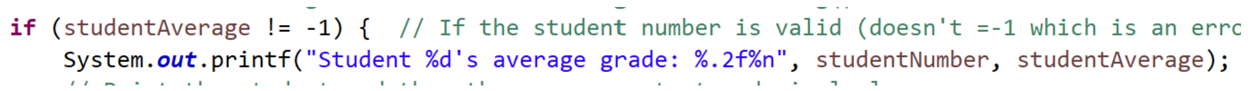
```
System.out.print("Enter student number (1-12): ")
studentNumber = scanner.nextInt();  // Make their
```

After this, I created a double variable called studentAverage which was equal to the number that came from the calculations in the studentAvg() method using the studentNumber parameter in the gradeBook object. (explained more in the GradeBook class reflection log)



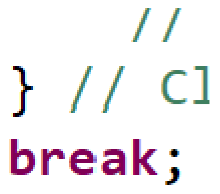
```
double studentAverage = gradeBook.studentAvg(studentNumber);
```

Next, I wrote an if statement saying if studentAverage isn't -1 to print the Student's average grade. (if it was -1 then that means the studentAvg() method sent back -1 to indicate an error, so it would re-prompts the user), if it isn't equal to -1 then I print the "Student (studentNumber)'s average grade is: (studentAverage rounded to 2 decimal places with %2f)". (%n is the same as \n but just is specific for printf statements)



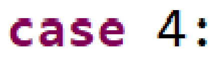
```
if (studentAverage != -1) {  // If the student number is valid (doesn't =-1 which is an errc
    System.out.printf("Student %d's average grade: %.2f%n", studentNumber, studentAverage);
```

Lastly, I closed the if statement and ended the case with a break.



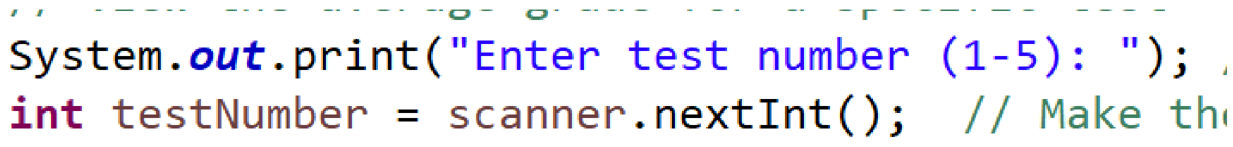
```
        //
} // Cl
break;
```

The next case I made was case 4.



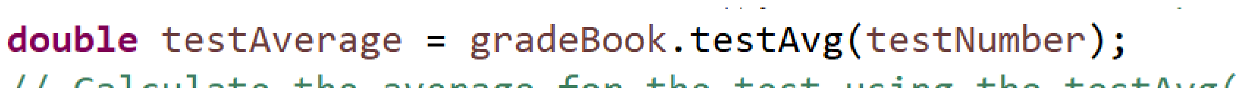
```
case 4:
```

I then prompted the user to enter which test they wanted to know the average of and saved this as the variable testNumber.



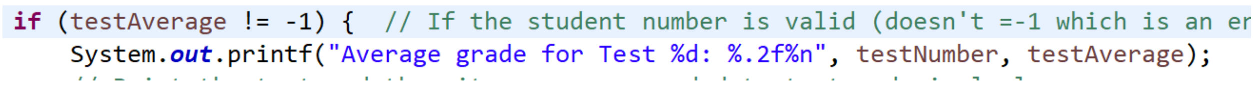
```
System.out.print("Enter test number (1-5): ");
int testNumber = scanner.nextInt();  // Make th
```

After this, I created a double variable called testAverage which was equal to the number that came from the calculations in the testAvg() method using the testNumber parameter in the gradeBook object. (explained more in the GradeBook class reflection log)



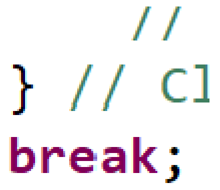
```
double testAverage = gradeBook.testAvg(testNumber);
// Calculate the average for the test using the testAvg(
```

Next, I wrote an if statement saying if testAverage isn't -1 to print the test's average grade. (if it was -1 then that means the testAvg() method sent back -1 to indicate an error, so it would re-prompts the user), if it isn't equal to -1 then I print the “Average grade for Test (testnumber): (testAverage rounded to 2 decimal places with %2f)”. (%n is the same as \n but just is specific for printf statements)



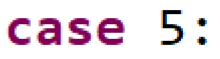
```
if (testAverage != -1) {  // If the student number is valid (doesn't =-1 which is an er
    System.out.printf("Average grade for Test %d: %.2f%n", testNumber, testAverage);
```

Lastly, I closed the if statement and ended the case with a break.



```
        //
} // Cl
break;
```

The next case I made was case 5.



```
case 5:
```

This was for if the user wanted to quit the program. I message indicating that the program is stopping would be printed, I then wrote a break line to end the case

```
System.out.println("Exiting the Grade Book Application. Thank you! =)");
break; // End case 5
```

The last case I made was the default case.

```
default:
```

This case was for if choice 3 (student average) or choice 4 (test average) returned -1, this error message would then show up prompting the user to try again.

```
System.out.println("Invalid option. Please try again.");
```

I then closed the switch statement with a brace.

```
}
```

And closed the do-while loop making the condition for quitting if choice = 5 (the quit option)

```
} while (choice != 5);
```

I then close the scanner with the .close() method. This releases the system.in resources that are used for inputs from your console (keyboard). It is not super necessary in this code as no errors would occur if it wasn't there, however after reading some comments from StackOverflow and W3schools it became clear that this is a good common practice for future longer and more complex programs.

```
scanner.close();
```

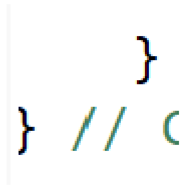
Lastly, I closed the main method with a brace.

```
}
```

And the public class CourseGrades with a brace.

```
}
```

Together they looked like this.

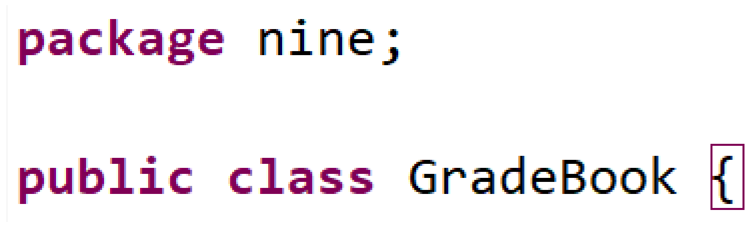


```
    }
} // (
```

GradeBook

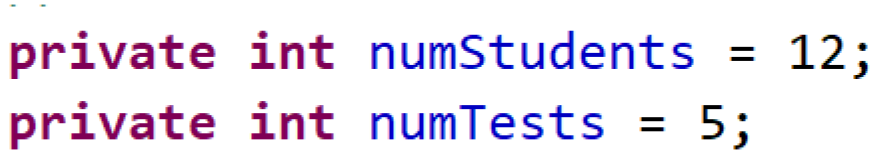
(the package this file is in is Mastery)(NOTE: since I did this at home and my eclipse isn't linked my package is called nine but in the code I submit it will be Mastery)

The public class is declared as GradeBook(It's declared as public so the CourseGrades class can access its information and return information).



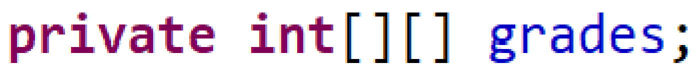
```
package nine;

public class GradeBook {
```

I then create two private integer variables for the number of students and tests. I set the variable numStudents to 12 and the variable numTests to 5.



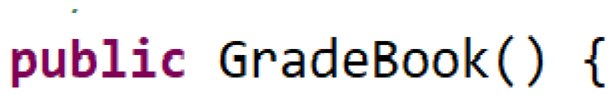
```
private int numStudents = 12;
private int numTests = 5;
```

After this I created a private two dimensional array called grades that would hold the grades of all of the students (rows) for all of the tests (columns) as integers.



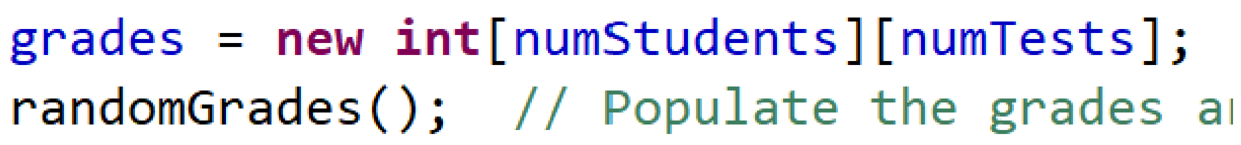
```
private int[][] grades;
```

Next I made the public constructor GradBook() (you can tell its a constructor because it has the same name as the class)



```
public GradeBook() {
```

I then initialized the grades array with 12 student and 5 test (from the variables created at the start), and then populated the array with randomly generated numbers from the randomGrades() method.

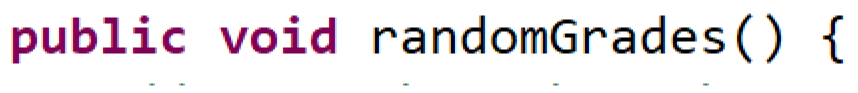


```
grades = new int[numStudents][numTests];
randomGrades();  // Populate the grades a
```

After this I closed the constructor with a brace.

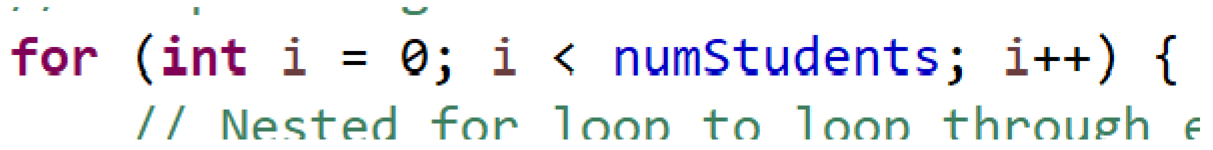
```
}
```

Next I made the public void method randomGrades.



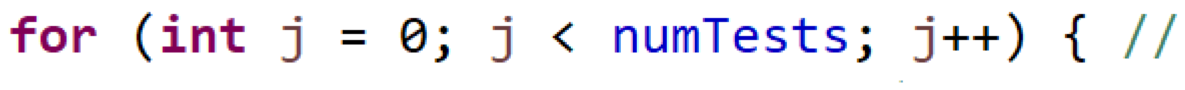
```
public void randomGrades() {
```

The first for loop in this method goes through each of the 12 students, hence why it stops at numStudents(12). (note: i = 0 because arrays are 0 based), 1 gets added to "i" each time to move onto the next student.



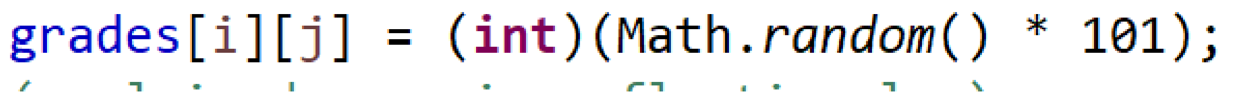
```
for (int i = 0; i < numStudents; i++) {
    // Nested for loop to loop through
```

The nested for loop in randomGrades was for the randomly generated test grades. As you can see it goes through each of the 5 tests and stops when "j" equals numTests (5) (because array's are 0 based). I then add 1 to "j" to move onto the next test. (this nested for loop effectively goes through all 5 tests for all 12 students)



```
for (int j = 0; j < numTests; j++) { //
```

In this nested for loop I make the grades array row equal to the "i" (student) and the column equal to "j" (test). To randomly generate a number for the test I use math random to generate a number between 0-1 and multiply it by 101 (so it includes 100) and make it an integer because Java always rounds down so 101 won't be possible to get.



```
grades[i][j] = (int)(Math.random() * 101);
```

I then close the nested for loop with a brace.

```
}
```

The outer for loop with a brace.

```
}
```

And the randomGrades() method with a brace.

```
}
```

Together they look like this.

```
            }
        } // C
    } // Close
```

After this I created the public void method getgrades() for if the user selected choice 1 from the menu in CourseGrades. (I passed this method the integer variable studentNumber from the CourseGrades class)

```
public void getGrades(int studentNumber) {
```

I started this method by making an error checker. If the user entered studentNumber was not within the range (if it’s less than 1 or more than 12) than an error message will be printed, and the method will be exited sending the user back to the menu.

```
if (studentNumber < 1 || studentNumber > numStudents) {
    System.out.println("Invalid student number."); // Pr
    return;  // Exit the method (will lead to you going
}
```

If the user’s answer was valid then the user would be prompted to enter the test grades for the student they chose.

```
System.out.println("Enter grades for Student " + studentNumber);
```

I then made a for loop to ensure each of the 5 tests was run through (i = 0 because arrays are 0 based), once i is equal to numTests the loop stops because 0-4 is 5 tests so it stops at 5. I then add 1 to "i" to move onto the next test.

```java
for (int i = 0; i < numTests; i++) {
```

Next, I prompt the user to enter the grade for the test there currently on (I use i + 1 because the user is using 1 based not 0 based)

```java
System.out.print("Enter grade for Test " + (i + 1) + ": ");
```

Their input is then saved in the test spot (column) for the specific student (row) of the grades array through this line.

```java
grades[studentNumber - 1][i] = new java.util.Scanner(System.in).nextInt();
```

I then close the for loop with a brace.

```java
}
```

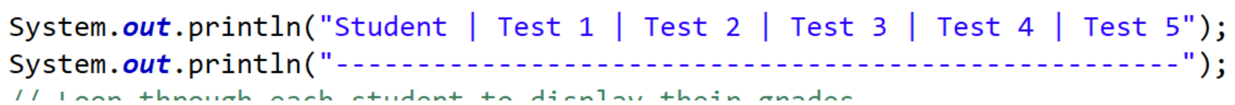
And close the method getGrades() with a brace.

```java
}
```

I then added the next method public voidshowGrades() for if the user chose the second choice to show all of the students, tests, and grades.

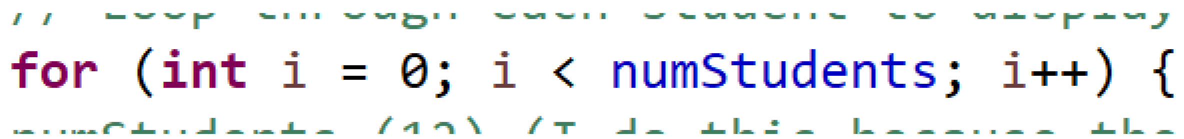
```java
public void showGrades() {
    // Print a header row f
```

Next, I print the header with the students column and the 5 test columns. There is a line below to separate the header and the actual information.



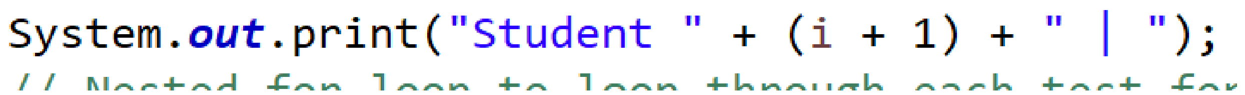
```
System.out.println("Student | Test 1 | Test 2 | Test 3 | Test 4 | Test 5");
System.out.println("--------------------------------------------------");
```

After, I begin a for loop to loop through all 12 students (it stops at 12 because array's and now the index is 0 based), I then add 1 to "i" each iteration to move onto the next student.



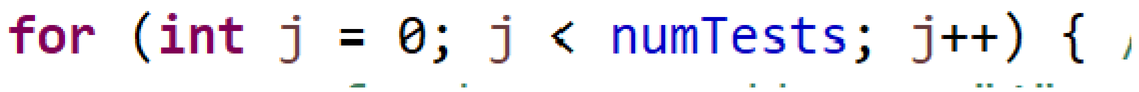
```
for (int i = 0; i < numStudents; i++) {
```

I then print the student and their number which is i + 1 to be 1 based instead of 0 based. There is a " | " to separate the student and the tests. (note: this loop prints the student (first part of the row) and then the next nested loop prints that students test in that row)



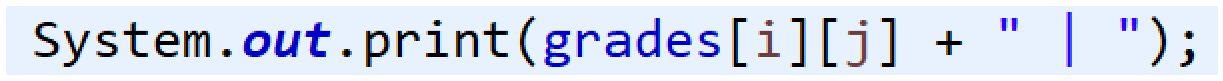
```
System.out.print("Student " + (i + 1) + " | ");
```

Next, I create the nested for loop with the int variable "j" initialized as 0 to be 0 based for the array. As long as it's less than numTests(5) it will keep running (because 0-4 is the 5 tests). I then add 1 to move onto the next test "j" for the student "i".



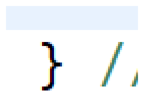
```
for (int j = 0; j < numTests; j++) {
```

After this I print the current student "i", and ONE of their tests followed by " | " to separate the tests (it prints one because in the next iteration it will print the second test, and then third test etc.)



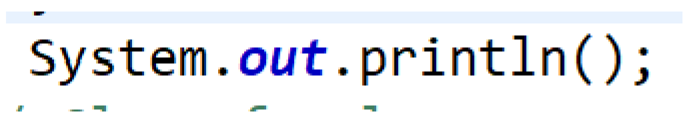
```
System.out.print(grades[i][j] + " | ");
```

I then close the nested for loop with a brace.



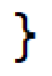
```
}
```

And print a space to move onto the next line for the next student (this is in the first for loop)



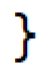
```
System.out.println();
```

I then close the first for loop with a brace.



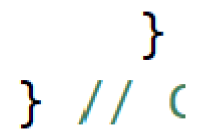
```
}
```

And close the showGrades() method with a brace.



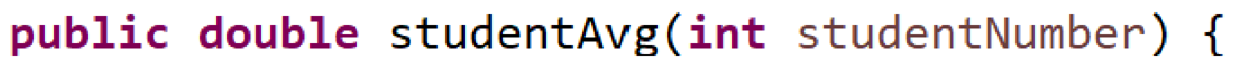
```
}
```

Together they looked like this



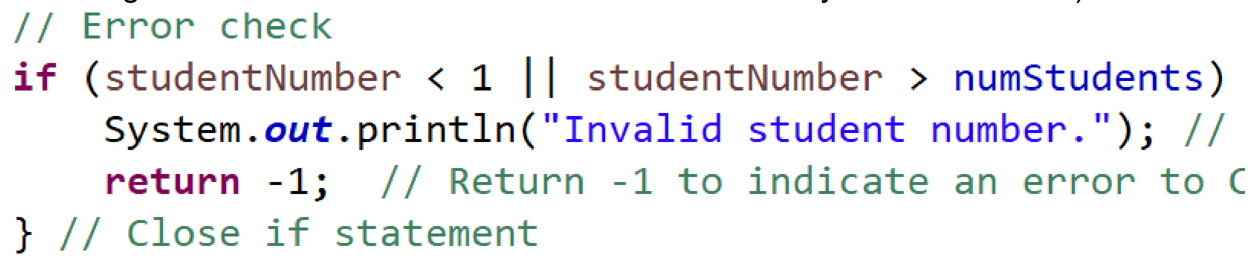
```
        }
} // (
```

Next I created the public double method studentAvg() that was passed the parameter studentNumber from the CourseGrads class.



```
public double studentAvg(int studentNumber) {
```

I began this code with an error checker. If the studentNumber entered by the user was less then 1 or bigger then numStudents(12) then an error message would be printed and -1 would be returned (this doesn't really do anything but since it's not a void method something needs to be returned and -1 doesn't activate any code so it works)



```
// Error check
if (studentNumber < 1 || studentNumber > numStudents)
    System.out.println("Invalid student number."); //
    return -1;  // Return -1 to indicate an error to C
} // Close if statement
```

I then initialize an int variable total as 0 outside of the for loop to come, because multiple iterations will occur.

```java
int total = 0;
```

After this, I create the for loop that will go through each of the students' tests. “i” is initialized as 0 to match the 0 based array, and the loop will keep running until “i” is equal to numTests(5) (stops at 5 because 0-4 is 5 tests), I then add 1 onto “i” to move onto the next test.

```java
for (int i = 0; i < numTests; i++) {
```

Here I add the current test grades “i” that the loop is on for the student -1 (because you -1 from student number which is 1 based so the grades array (0 based) can find the right student), and I get this from the two dimensional grades array. I add this to the total variable declared outside the loop (each iterations grades will be added giving the total of all the test combined)

```java
total += grades[studentNumber - 1][i];
```

I then close the for loop with a brace.

```java
}
//
```

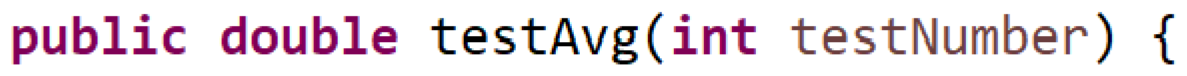
And return the now double variable total / by numTests(5) to get the students average. This is returned to CourseGradse to be printed for the user to see.

```java
return (double) total / numTests;
```

I then close the studentAvg() method with a brace.

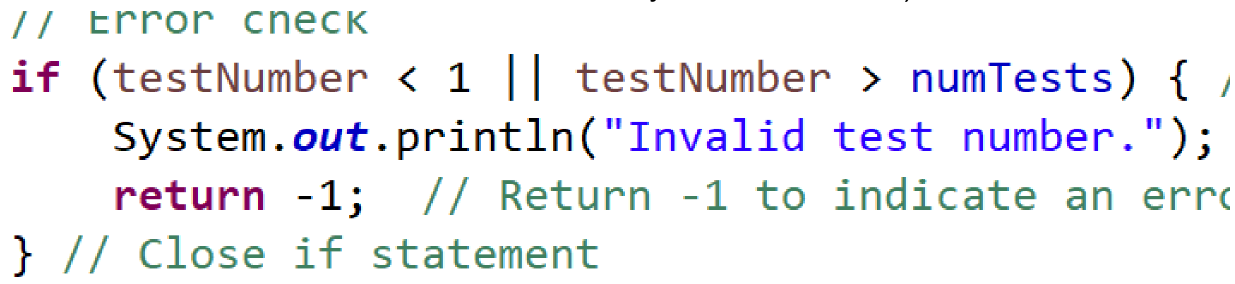
```java
} /
```

The last method I make is the public double teseAvg() method that I pass the parameter testNumber from CourseGrades.



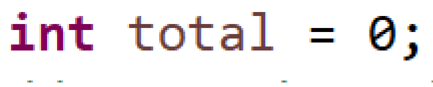
```
public double testAvg(int testNumber) {
```

I began this code with an error checker. If the testNumber entered by the user was less than 1 or bigger than numTests(5) then an error message would be printed and -1 would be returned (this doesn't really do anything but since it's not a void method something needs to be returned and -1 doesn't activate any code so it works)



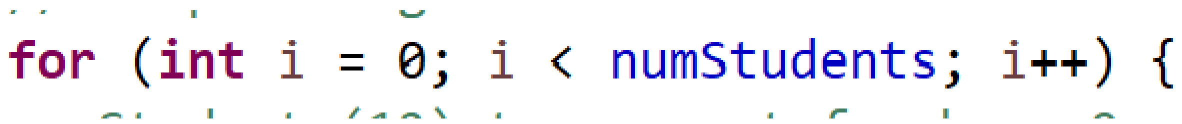
```
// Error check
if (testNumber < 1 || testNumber > numTests) { /
    System.out.println("Invalid test number.");
    return -1;  // Return -1 to indicate an err
} // Close if statement
```

I then initialize an int variable total as 0 outside of the for loop to come, because multiple iterations will occur.



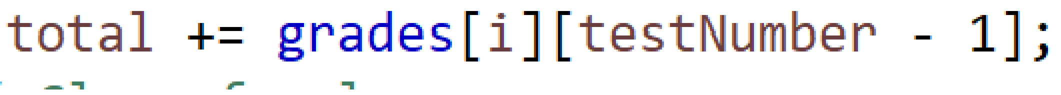
```
int total = 0;
```

After this, I create the for loop that will go through each of the students' for a test. “i” is initialized as 0 to match the 0 based array, and the loop will keep running until “i” is equal to numStudents(12) (stops at 12 because 0-11 is 12 students), I then add 1 onto “i” to move onto the next student.



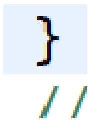
```
for (int i = 0; i < numStudents; i++) {
```

Here I add the current student's test grades “i” that the method is finding the average test grade of testNumber - 1(because you -1 from test number which is 1 based so the grades array (0 based) can find the right test), and I get this from the two dimensional grades array. I add this to the total variable declared outside the loop (each iterations student's tests will be added giving the total of all the student's combined)



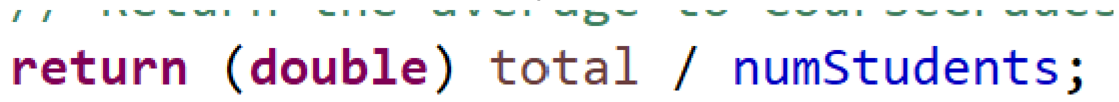
```
total += grades[i][testNumber - 1];
```

I then close the for loop with a brace.



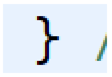
```
}
//
```

And return the now double variable total / by numStudents(12) to get the tests average. This is returned to CourseGradse to be printed for the user to see.



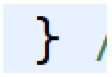
```
return (double) total / numStudents;
```

I then close the testAvg() method with a brace.



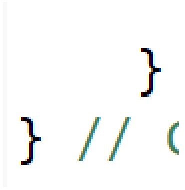
```
} /
```

And close the public GradeBook class with a brace.



```
} /
```

Together they look like this.



```
    }
} // (
```