CMM024 Object Oriented Programming

Practical Session: 4

In the lecture we went through a few coding examples about creating, assigning values and using Arrays and ArrayLists

1. Arrays: creating and Initializing an Array

Brief:

In this exercise, you must create an array to store **n** integer values. You must use the Math.random(…) method, which return a double from 0.0 to 1.0 (1.0 excluded) to initialise these values, to finally print these values in the terminal console. Note that you will have to typecast the obtained values to integers as this is what the array expects.

Overall Tasks:

1) Create a **Lab4_1.java** java file, and create a **main** function

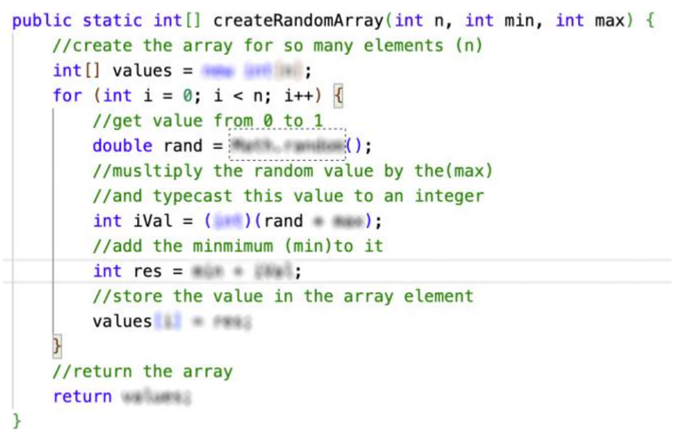
2) Create the code described above.

3) Run the program so the output is like the one shown below.

Code hints

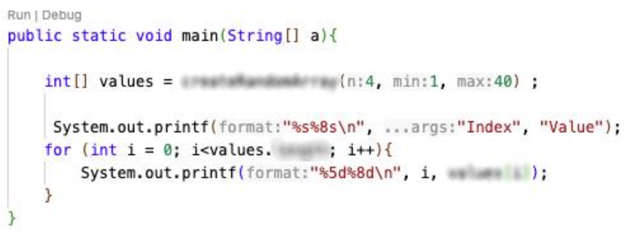
Structure your code as shown below and replace the blurred content with the actual code.

Method to create



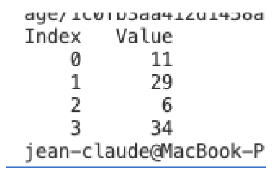
```java
public static int[] createRandomArray(int n, int min, int max) {
    //create the array for so many elements (n)
    int[] values = [illegible];
    for (int i = 0; i < n; i++) {
        //get value from 0 to 1
        double rand = [illegible]();
        //musltiply the random value by the(max)
        //and typecast this value to an integer
        int iVal = ([illegible])(rand [illegible]);
        //add the minmimum (min)to it
        int res = [illegible];
        //store the value in the array element
        values[illegible]
    }
    //return the array
    return [illegible]
}
```

The Main method to code



```
Run | Debug
public static void main(String[] a){

    int[] values = [redacted](n:4, min:1, max:40) ;

     System.out.printf(format:"%s%8s\n", ...args:"Index", "Value");
    for (int i = 0; i<values.[redacted]; i++){
        System.out.printf(format:"%5d%8d\n", i, [redacted]);
    }
}
```

Output:



```
age/1c01b3aa412d1438a
Index   Value
    0      11
    1      29
    2       6
    3      34
jean-claude@MacBook-P
```

2. Arrays: assigning values to an array by invoking a method

Brief:

Create an array where each of the assigned value is the previous one multiplied by a specific ratio. For instance, the series 1, 3, 9, 27 has a ratio (r) of 3, with a starting value (a) of 1. The number of values (n) is 4.

Overall Tasks:

1) Create a **Lab4_2.java** java file, and create a **main** function
2) Create the code described below in the main method.
3) There are 3 tasks to complete
4) Run the program so the output is like the one shown below.

Task 1:

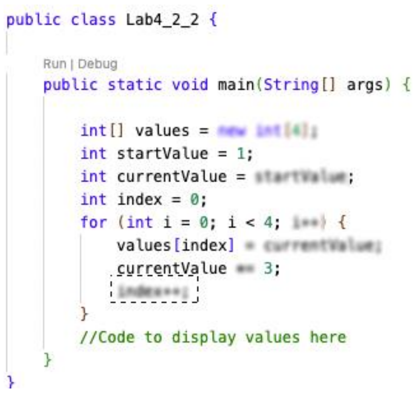
Pseudo Code

```
Create an array of 4 integers and name it as “values”
Starting value (Integer) = 1;
Current value (Integer) = Starting value
Array Index (Integer) = 0;
For (int i = 0; i < 4; i++) {
        Array values[index] = Current Value;
        Current Value = Current value * 3;
        Increase index
}
Print array Values
```

Make sure the code above(and below) produces the 1,3,9,27 series.

Code hints

Note that the code below does not display the values. This code will be added after that for loop.



```
public class Lab4_2_2 {

    Run | Debug
    public static void main(String[] args) {

        int[] values = [redacted]
        int startValue = 1;
        int currentValue = [redacted];
        int index = 0;
        for (int i = 0; i < 4; [redacted]) {
            values[index] [redacted]
            currentValue [redacted] 3;
            [redacted]
        }
        //Code to display values here
    }
}
```

Task 2:

1) Still in the **Lab4_2.java** java file

2) Port the code you created in the main method to its own method.

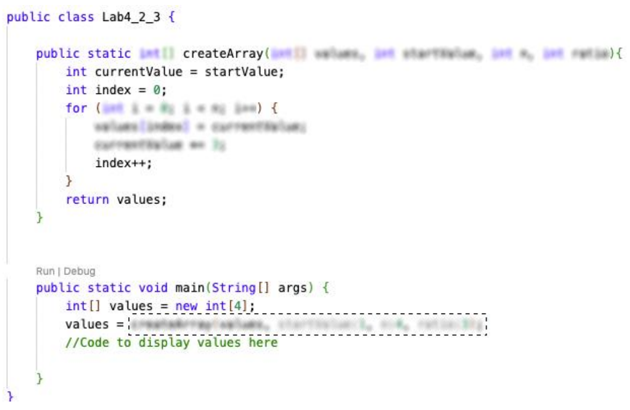
We have now the code working completed in Task 1, it is time to extract the relevant code lines and create a method that will perform this task. That function will need parameters.

- The array
- The starting value (a)
- The number of terms (n)
- The ratio (r)

public static int[] createArray(array, start value, number of values, ratio)

Code Hints

Method to create as well as the Main method to code



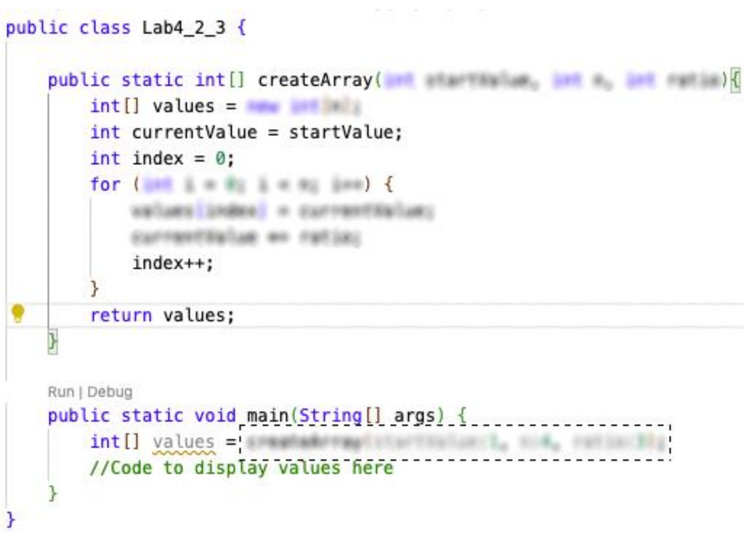
```
public class Lab4_2_3 {

    public static [redacted] createArray([redacted]){
        int currentValue = startValue;
        int index = 0;
        for ([redacted]) {
            [redacted]
            index++;
        }
        return values;
    }

    Run | Debug
    public static void main(String[] args) {
        int[] values = new int[4];
        values = [redacted]
        //Code to display values here

    }
}
```

Task 3:

Finally, we can ask the method to create the array as well as assigning the values for it. This means adding some extra code to create the array in the method, remove this code from the main, and off course removing the unneeded parameter in the method pertaining to array.

```
public static int[] createArray(start value, number of values, ratio){
```

Code Hints



```
public class Lab4_2_3 {

    public static int[] createArray([redacted]){
        int[] values = [redacted]
        int currentValue = startValue;
        int index = 0;
        for ([redacted]) {
            [redacted]
            index++;
        }
        return values;
    }

    Run | Debug
    public static void main(String[] args) {
        int[] values = [redacted]
        //Code to display values here
    }
}
```

As you can see the code in the main method has drastically decrease. The fortunate outcome is that this method can be re-used in another project!

3. Arrays: performing tasks on arrays

Brief:

In this exercise you will perform some tasks on arrays such as rotating array values from left to right and vice versa, swapping element values, displaying content of an array.
You will first code the solution for each of the task in the main method and thereafter transfer that code in their own method.
We will first use the method created in the previous example to generate the value.
Then you will code a method to generate random values instead.

Overall Tasks:

1) Create a **Lab4_4.java** java file, and create a **main** function

2) Paste and copy the ***createArray*** method (Generating a geometric series) you created in the last exercise.

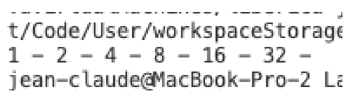
3) Create the code described below.

4) Run the program so the output is like the one shown below.

Task 1:

In the main method, Write code to generate a series with 6 terms with 1 as a start value and a ratio of 2. Use the method you already created.

<u>*Output*</u>



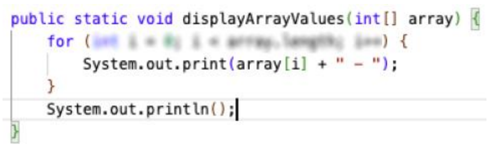
```
t/Code/User/workspaceStorage
1 - 2 - 4 - 8 - 16 - 32 -
jean-claude@MacBook-Pro-2 La
```

Task 2:

- Write code to display the values contained in an array.
- Display content of the array created above, by adding code in the main method.

<u>*Code Hints*</u>

Method to code to display the elements of an array. Make sure you write the code that is blurred.



```java
public static void displayArrayValues(int[] array) {
    for ([illegible]) {
        System.out.print(array[i] + " - ");
    }
    System.out.println();
}
```

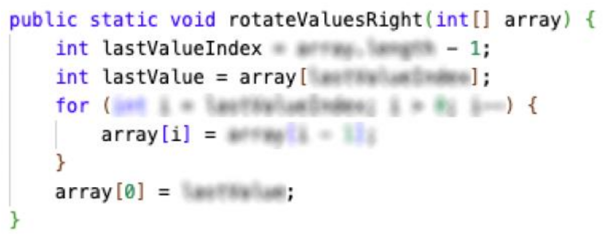
Task 3

- Write code to rotate the value in the array from left to right
- Display content of the array after rotation.

<u>*Pseudo Code*</u>

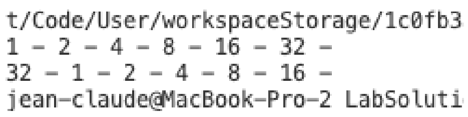
```
Save the last term index into a variable (Remember that an array can have any length)
Save the last value referred to by this last term index
  // index > bigger than 0 means that as index decreases, the for loop will only run
  //when that index is positive and not equal to zero
For( index = last term index,  index > bigger than 0,  index--){
       Value [index] = previous value [index - 1]
}
First value[0] = last value
```

<u>*Code hint*</u>



```java
public static void rotateValuesRight(int[] array) {
    int lastValueIndex [illegible] - 1;
    int lastValue = array[[illegible]];
    for ([illegible]) {
        array[i] = [illegible]
    }
    array[0] = [illegible];
}
```

Output



```
t/Code/User/workspaceStorage/1c0fb3
1 - 2 - 4 - 8 - 16 - 32 -
32 - 1 - 2 - 4 - 8 - 16 -
jean-claude@MacBook-Pro-2 LabSoluti
```

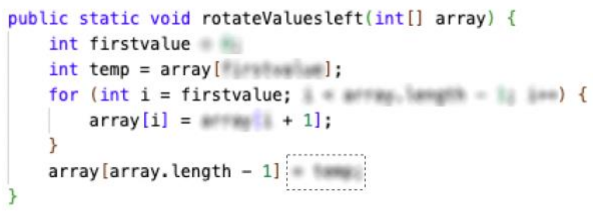
Task 4:

- Write code to rotate the value in the array from right to left
- Display content of the array after rotation.

Pseudo Code

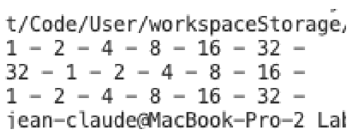
```
Save the first term index into a variable (Note: an array first value is indexed as 0)
Save the first value referred to by this first term index
 // index < array length means that as index increases, the for loop will only run
 //when that index is small the length of the array.
For( index = first term index,  index < array length,  index++){
       Value [index] = next value [index + 1]
}
Last value[last value index] = first value
```

Code hint



```
public static void rotateValuesleft(int[] array) {
    int firstvalue [redacted]
    int temp = array[[redacted]];
    for (int i = firstvalue; [redacted]) {
        array[i] = [redacted] + 1];
    }
    array[array.length - 1] [redacted]
}
```

Output



```
t/Code/User/workspaceStorage,
1 - 2 - 4 - 8 - 16 - 32 -
32 - 1 - 2 - 4 - 8 - 16 -
1 - 2 - 4 - 8 - 16 - 32 -
jean-claude@MacBook-Pro-2 Lal
```

Task 5:

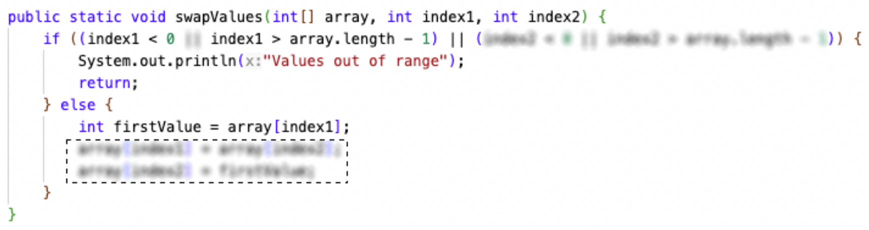
- Write code to swap one term of an array indexed by *index1* for the term of an array indexed by *index2*.
- Display content of the array after rotation.

Pseudo Code

```
Check if index1 and index2 are < 0 and not bigger than the length of the array
If indexes are not right{
       Display an error message
```

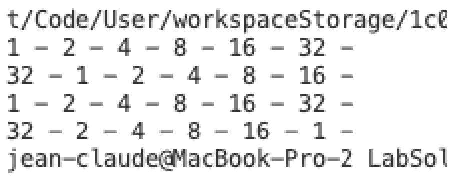
```
            Force return
    }
    Else{
            save value referred to by first index in a variable
            the value referred to by index1 equals the value referred to by index2
            the value referred to by index2 equals the saved value
    }
```

Code hints



```
public static void swapValues(int[] array, int index1, int index2) {
    if ((index1 < 0 || index1 > array.length - 1) || ([redacted])) {
        System.out.println(x:"Values out of range");
        return;
    } else {
        int firstValue = array[index1];
        [redacted]
    }
}
```

Output



```
t/Code/User/workspaceStorage/1c
1 - 2 - 4 - 8 - 16 - 32 -
32 - 1 - 2 - 4 - 8 - 16 -
1 - 2 - 4 - 8 - 16 - 32 -
32 - 2 - 4 - 8 - 16 - 1 -
jean-claude@MacBook-Pro-2 LabSol
```

4. (Advanced) Using arrays: a real example

Brief:

Often you may need to perform some maths when coding. Your only information might be an equation.
The brief is to create a geometric series with the values stored in an array. The objective is to sum up all the elements in that array, first by iterating through the array element, and then using a formula to perform that operation.
This was covered in the advanced exercises in session 3 when we dealt with compounded loans etc.
Let's take this example, 1, 3, 9, 27. In this series we have 4 terms "n", the starting value "a" is 1 and the ratio "r" between each number is 3. i.e., 1, 1*3, 3*3, 9*3 etc.

Notes:

Use the method create in Task 2. i.e.

```
public static int[] createArray(int a, int n, int r)
```

Here is some extra information for the students who have a curious nature.
A geometric series is the sum of an "n" of terms that have a constant ratio between successive terms.

$$sum = ar^0 + ar^1 + ar^2 + ar^3 + \cdots + ar^{n-1}.$$

$$sum = 1 * 3^0 + 1 * 3^1 + 1 * 3^2 + 1 * 3^3 = 40$$

Just for information it can be written as:

$$sum = ar^{k=0} + ar^{k=1} + ar^{k=2} + ar^{k=3} + \cdots + ar^{k=(n-1)}$$

$$sum = \sum_{k=0}^{n-1} ar^k$$

To get the sum, you can create some code that will sum up each term until the last one, or you can use the formula below.

$$sum = a * \left(\frac{r^n - 1}{r - 1}\right)$$

We have seen that last week.

Overall Tasks

1) Create a **Lab4_Sums.java** java file, and create a **main** function
2) Paste and copy the ***createArray*** method (Generating a geometric series) you created in the last exercise.
3) Create the code described below.
4) Test your code and display the results etc.

Task 1

Sum up the terms of any array using the traditional **for loop** method in a method.

```
public static int sumArray(int[] values)
```

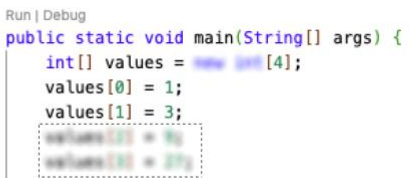
Create a method to display the content of this array, including the sum

```
public static void printArrayElements(int[] values)
```

Test your code using the , 1, 3, 9, 27 values;

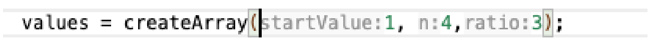
Note

To create the series of value, you create an array of 4 values and assign a value to each of the element.



```
Run | Debug
public static void main(String[] args) {
    int[] values = [illegible][4];
    values[0] = 1;
    values[1] = 3;
    [illegible]
```

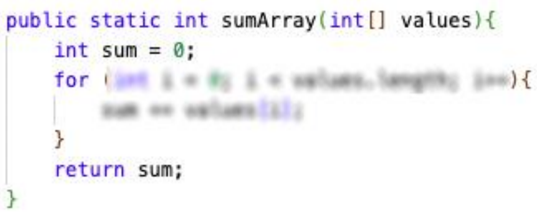
Or use the method you created in Task 2.



```
values = createArray(startValue:1, n:4,ratio:3);
```

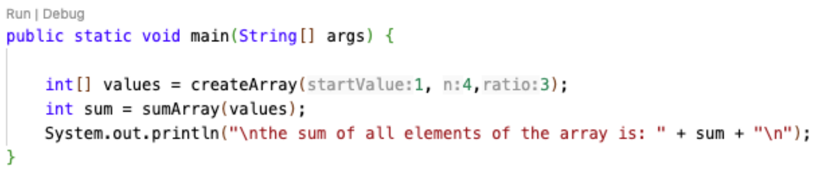
Code hints

The method



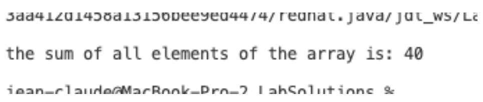
```
public static int sumArray(int[] values){
    int sum = 0;
    for ([redacted]){
        [redacted]
    }
    return sum;
}
```

The main method



```
Run | Debug
public static void main(String[] args) {

    int[] values = createArray(startValue:1, n:4,ratio:3);
    int sum = sumArray(values);
    System.out.println("\nthe sum of all elements of the array is: " + sum + "\n");
}
```

Output



```
5aa412d1458a13156beeyed4474/redhat.java/jdt_ws/La

the sum of all elements of the array is: 40

jean-claude@MacBook-Pro-2 LabSolutions %
```

Task 2

In task 1 , you have sum up the elements of the array by using a for loop and iterate through the elements to sum them up. Here you are going to use a formula that does the same but is faster when the length gets long. Also note that when using the formula, we do not need to create the array in the first place! i.e. instead of using the values, we use the properties.

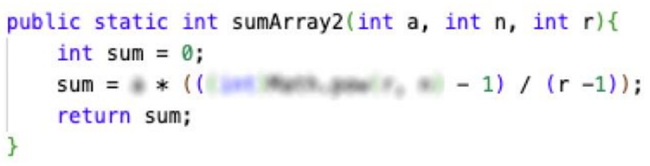
Create a method to sum up elements of an array using the formula

```
public static int sumArray2(int a, int n, int r)
```

Create a method to display all details of an array

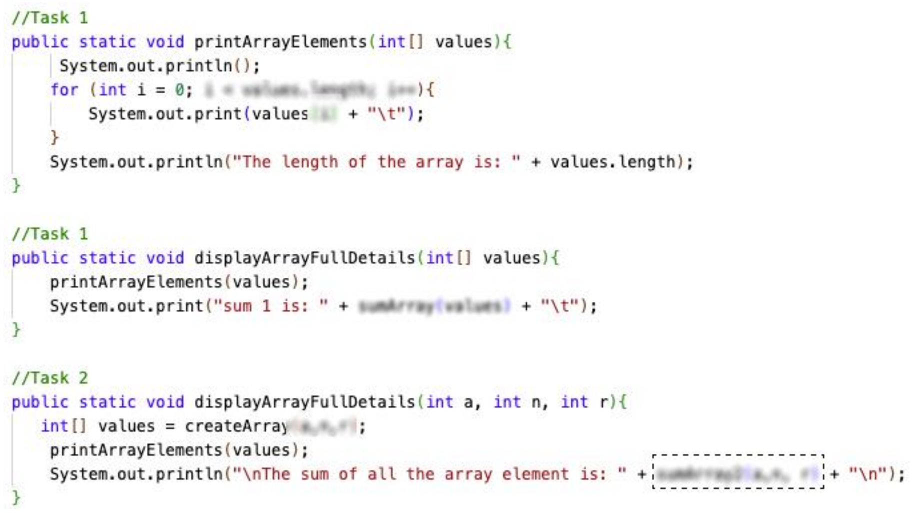
Code hints

The method



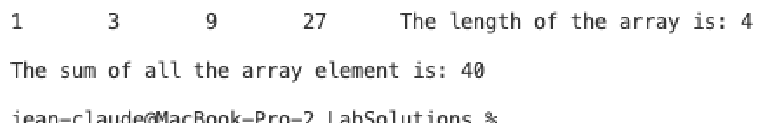
```
public static int sumArray2(int a, int n, int r){
    int sum = 0;
    sum = [redacted] * (([redacted] - 1) / (r -1));
    return sum;
}
```

The displaying methods



```
//Task 1
public static void printArrayElements(int[] values){
    System.out.println();
    for (int i = 0; [redacted]){
        System.out.print(values[redacted] + "\t");
    }
    System.out.println("The length of the array is: " + values.length);
}

//Task 1
public static void displayArrayFullDetails(int[] values){
    printArrayElements(values);
    System.out.print("sum 1 is: " + [redacted] + "\t");
}

//Task 2
public static void displayArrayFullDetails(int a, int n, int r){
    int[] values = createArray[redacted];
    printArrayElements(values);
    System.out.println("\nThe sum of all the array element is: " + [redacted]+ "\n");
}
```

Output



```
1       3       9       27      The length of the array is: 4

The sum of all the array element is: 40

jean-claude@MacBook-Pro-2 LabSolutions %
```

5. ArrayLists: performing tasks on ArrayLists

Brief:

The purpose of this series of exercise is to develop a program to store grades that you input from the console, then finding the lowest and highest grades from that list.

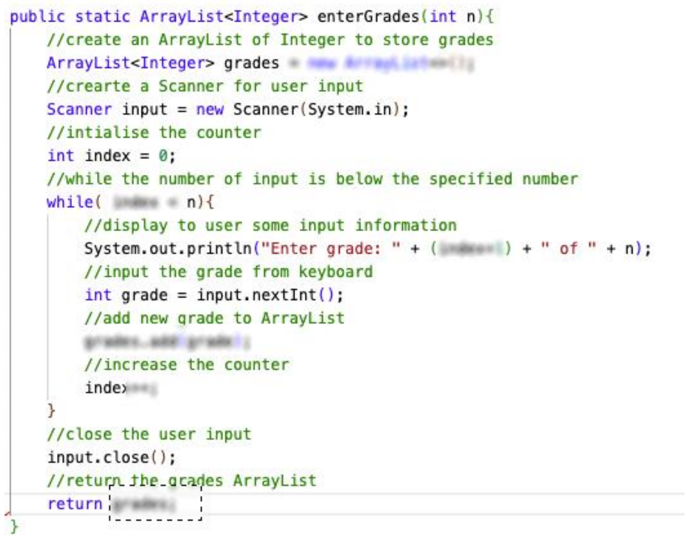
Overall Tasks

- Create a java file called Grading.java
- Code the tasks below
- Run your code to test if it is right

Task 1

Create a method that will allow a user to input so many grades (n) and store them into an ArrayList, which is returned by this method.
Test that method to ensure it is working properly.

Code Hints

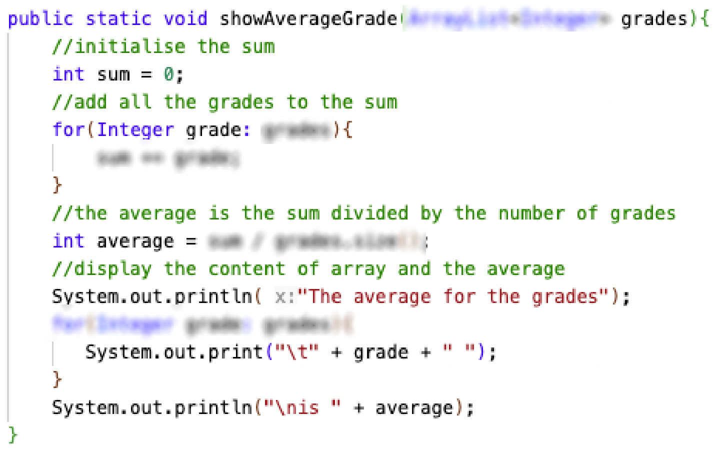


```
public static ArrayList<Integer> enterGrades(int n){
    //create an ArrayList of Integer to store grades
    ArrayList<Integer> grades [redacted]
    //crearte a Scanner for user input
    Scanner input = new Scanner(System.in);
    //intialise the counter
    int index = 0;
    //while the number of input is below the specified number
    while( [redacted] n){
        //display to user some input information
        System.out.println("Enter grade: " + ([redacted]) + " of " + n);
        //input the grade from keyboard
        int grade = input.nextInt();
        //add new grade to ArrayList
        [redacted]
        //increase the counter
        index[redacted]
    }
    //close the user input
    input.close();
    //return the grades ArrayList
    return [redacted]
}
```

Task 2

Add and create a method that displays the average grade from a list of grades
Test that method to ensure it is working properly.

Code Hints

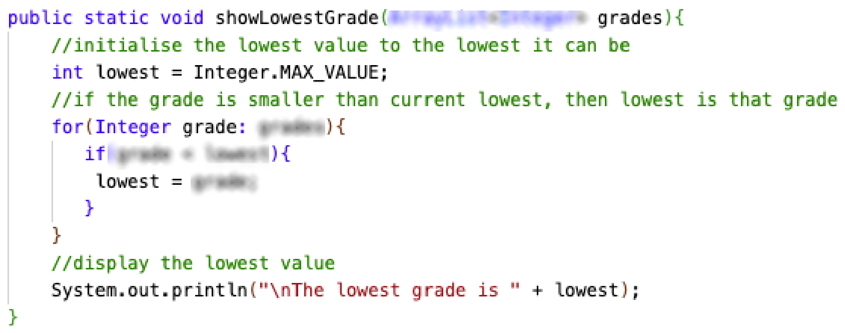


```
public static void showAverageGrade([redacted] grades){
    //initialise the sum
    int sum = 0;
    //add all the grades to the sum
    for(Integer grade: [redacted]){
        [redacted]
    }
    //the average is the sum divided by the number of grades
    int average = [redacted]
    //display the content of array and the average
    System.out.println( x:"The average for the grades");
    [redacted]
        System.out.print("\t" + grade + " ");
    }
    System.out.println("\nis " + average);
}
```

Task 3

Add and create a method that displays the lowest grade from a list of grades
Test that method to ensure it is working properly.

Code Hints

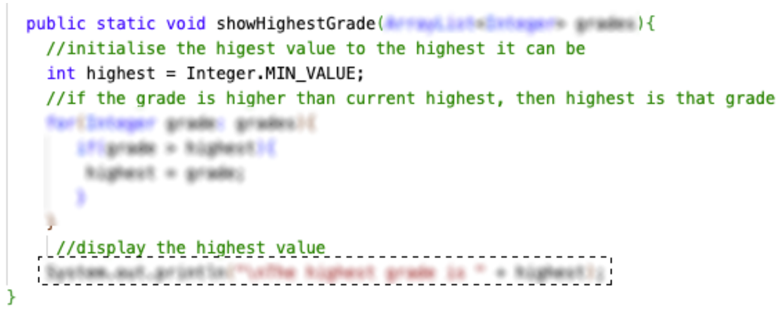


```
public static void showLowestGrade([illegible] grades){
    //initialise the lowest value to the lowest it can be
    int lowest = Integer.MAX_VALUE;
    //if the grade is smaller than current lowest, then lowest is that grade
    for(Integer grade: [illegible]){
        if[illegible]){
         lowest = [illegible]
        }
    }
    //display the lowest value
    System.out.println("\nThe lowest grade is " + lowest);
}
```

Task 4

Add and create a method that displays the highest grade from a list of grades
Test that method to ensure it is working properly.

Code Hints

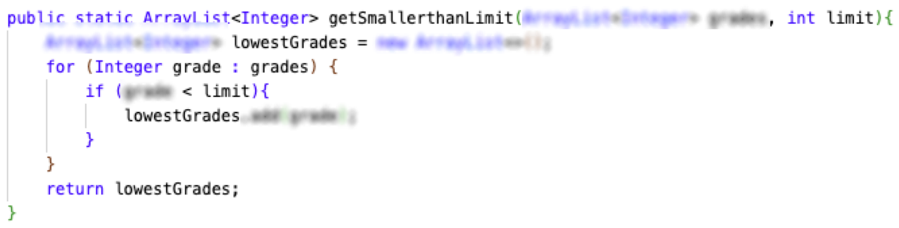


```
public static void showHighestGrade([illegible]){
  //initialise the higest value to the highest it can be
  int highest = Integer.MIN_VALUE;
  //if the grade is higher than current highest, then highest is that grade
  [illegible]
  //display the highest value
  [illegible]
}
```

Task 5

Add and Create a method that returns any grades that are smaller than a threshold (limit) and display.
Test that method to ensure it is working properly.

Code Hints

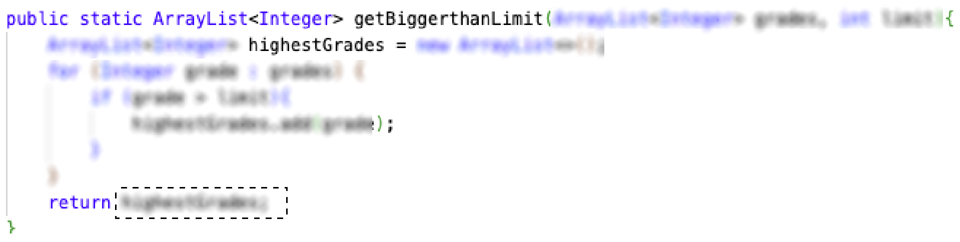


```
public static ArrayList<Integer> getSmallerthanLimit([illegible], int limit){
    [illegible] lowestGrades = [illegible]
    for (Integer grade : grades) {
        if ([illegible] < limit){
            lowestGrades[illegible]
        }
    }
    return lowestGrades;
}
```

Task 6

Add and Create a method that returns any grades that are higher than a threshold (limit) and display
Test that method to ensure it is working properly.

Code Hints

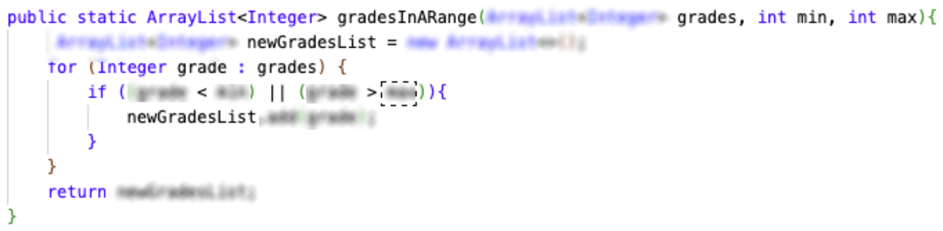


```java
public static ArrayList<Integer> getBiggerthanLimit([redacted]){
    [redacted] highestGrades = [redacted]
    [redacted]
        [redacted]
            [redacted]);
        [redacted]
    [redacted]
    return [redacted]
}
```

Task 7

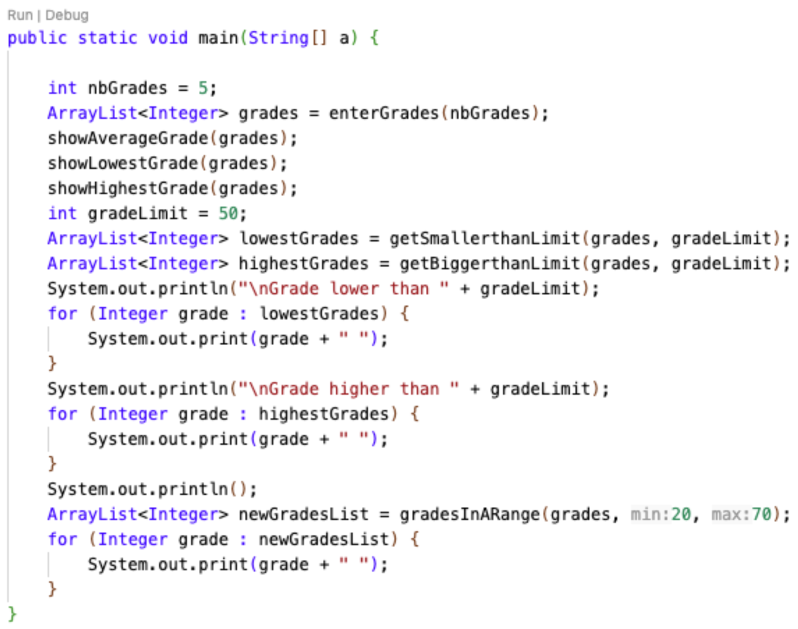
Add and Create a method that returns any grades that are lower than a minimum or higher than a maximum.

Code Hints



```java
public static ArrayList<Integer> gradesInARange([redacted] grades, int min, int max){
    [redacted] newGradesList = [redacted]
    for (Integer grade : grades) {
        if ([redacted] < [redacted]) || ([redacted] >[redacted])){
            newGradesList[redacted]
        }
    }
    return [redacted]
}
```

The Main code should look like this



```java
Run | Debug
public static void main(String[] a) {

    int nbGrades = 5;
    ArrayList<Integer> grades = enterGrades(nbGrades);
    showAverageGrade(grades);
    showLowestGrade(grades);
    showHighestGrade(grades);
    int gradeLimit = 50;
    ArrayList<Integer> lowestGrades = getSmallerthanLimit(grades, gradeLimit);
    ArrayList<Integer> highestGrades = getBiggerthanLimit(grades, gradeLimit);
    System.out.println("\nGrade lower than " + gradeLimit);
    for (Integer grade : lowestGrades) {
        System.out.print(grade + " ");
    }
    System.out.println("\nGrade higher than " + gradeLimit);
    for (Integer grade : highestGrades) {
        System.out.print(grade + " ");
    }
    System.out.println();
    ArrayList<Integer> newGradesList = gradesInARange(grades, min:20, max:70);
    for (Integer grade : newGradesList) {
        System.out.print(grade + " ");
    }
}
```