**CS 110 Programming Fundamentals I Lab 6: Files and methods**

In this lab you'll gain experience with Files, random numbers, and methods. What you learn in this lab should be helpful in completing homework #4, and in completing the final project (which requires the use of at least one File). Please submit your solutions to this lab to Canvas.

**As was mentioned in lecture, the solution to this lab will be posted to Canvas, but please attempt this lab before then, to get practice with Files and methods.**

**Part I – Reading Lines from a File, line-by-line**

Up to this point, you've learned in class about the modulus operator, running totals, as well as Java's file I/O capabilities. In the program that you will write, you will use all of these, to read a file line-by-line, and to output to the screen only certain portions of that file. The entire pseudocode of the program is given to in the file TerraceEverVerdant.java, that you can download from the course website. To complete this part of the lab:

1. Download the file *TerraceEverVerdant.java* from the course website, and save it to your computer.

2. Compile and run the java file. Because at this point the main method of your program contains just comments, your program will not perform any sort of task.

 Notice import java.util.Scanner; and import java.io.\*; at the top of the java file. As was mentioned in lecture, these important Java import statements provide functionality for reading input from the keyboard and reading from a file.

 Notice the throws IOException at the end of the main method declaration. As was discussed in lecture, this part of the code is required, if you are using Java's I/O capabilities.

1. Retrieve the *terraceEverVerdant.txt* file from Canvas, and save that file to your computer to the same location where you saved *TerraceEverVerdant.java*. Open the file, and inspect it. Can you make out any details?

2. Follow the instructions in the pseudocode in the *TerraceEverVerdant.java* file:

 Declare a variable, fileName, of type String, and assign it the value terraceEverVerdant.txt. Remember that Strings are enclosed by double quotes.

 Declare a variable, myFile, of type File, and assign it to be a new instance of the File class. Recall from lecture that to declare a new instance of a Class of type File, that references a specific *filename*, you would write:

File file = new File(filename);

 Declare a variable, inputFile, of type Scanner, and assign it to be a new instance of the Scanner class.

Give the Scanner class constructor the variable myFile as the single argument. As you've already done before several times, use the following template:

Scanner inputFile = new Scanner(file);

 Declare a variable, loopNum, of type int, and assign it the value 0

 Use a while loop, whose conditional is the following: inputFile.hasNext(). This while loop will iterate through its body as long as the reference to inputFile is not yet pointing to the end of the file. At each iteration of the while loop:

a. Increment loopNum by 1

b. Declare variable fileLine as a String, and assign it the value inputFile.nextLine()

c. Write an if statement, that checks if loopNum modulus 7 is equal to zero; if it is then use

System.out.println to print to the screen the value stored in fileLine.

 Close the file by issuing inputFile.close();

1. Compile and run your program. The output should print to the screen, every 7th line of the file, and the

ASCII image should be something that you can easily recognize.

**II. Generating Random Numbers**

As was shown in lecture, Java's Random class can be used to generate random numbers. In this part of the lab, you'll write a java program, that will prompt the user to input how many lottery numbers should be guessed. A few sample invocations of the program are shown in Figure 1.

I know you are busy, so I will guess your lottery numbers for you!

How many numbers should I guess? 3

What is the highest possible lottery number?

98

The first lottery Number is: 63

The next lottery Number is: 82

The last lottery Number is: 79

Good luck!

User indicates 3 numbers, no greater than 98

I know you are busy, so I will guess your lottery numbers for you!

How many numbers should I guess? 7

What is the highest possible lottery number?

7845

The first lottery Number is: 7175

The next lottery Number is: 2272

The next lottery Number is: 6106

The next lottery Number is: 903

The next lottery Number is: 3291

The next lottery Number is: 328

The last lottery Number is: 182

Good luck!

User indicates 7 numbers, no greater than 7845

I know you are busy, so I will guess your lottery numbers for you!

How many numbers should I guess? 5

What is the highest possible lottery number?

7634543

The first lottery Number is: 7590308

The next lottery Number is: 4283235

The next lottery Number is: 4609907

The next lottery Number is: 6447816

The last lottery Number is: 560108

Good luck!

User indicates 5 numbers, no greater than 7634543

**Figure 1:** Sample invocations of GenerateRandomLotteryNums.java

To complete this programming task:

1. Create a new file in jGRASP. Call it *GenerateRandomLotteryNums.java,* and save it.

2. The program uses the Scanner and Random classes, so you must import them at the top of the file:

import java.util.Scanner;

import java.util.Random;

3. Write a main method, that prints to the screen the following instructions (use a combination of println

and print statements, as necessary):

I know you are busy, so I will guess your lottery numbers for you!

How many numbers should I guess?

4. Set up a Scanner object; tell it to listen to the keyboard, by writng System.in in the constructor.

5. Declare a variable of type int, and use the nextInt() method of Scanner, to retrieve from the user, and save into the variable, an integer value.

6. Make your program print to the screen, the following message:

What is the highest possible lottery number?

7. Again use the nextInt() method of the Scanner object (you don't have to recreate a scanner object; just use the one that you created in the previous steps). Save the user's input into a new variable of type int, which will hold the maximum possible random number that should be generated. Be sure to use a good, descriptive variable name, such as highestPossibleLotteryNum.

8. In order to generate a random number, you must create the Random class in java, as was shown in lecture. To do this, you'd write:

Random randomNumbers = new Random();

9. Write a for loop, that will iterate from 0 through the value of the variable that holds the maximum possible random number that you want your program to generate.

for (int i = 0; i < numLotteryNums; i++)

10. Inside the body of your for loop:

◦ Create a new variable of type integer, and call it randomNum

◦ To generate a random number between zero and some upper-limit, use the nextInt() method of the Random class. Recall from lecture, that there are several forms of the nextInt method in the Random class. If you invoke the method without any arguments, then a random number in the range -2,147,483,648 to

+2,147,483,648 will be generated. If you pass a single integer argument with value n, then a random number in

the range 0 through n will be generated. For your purposes, generate a random number that is in the range of 0 through the value stored in highestPossibleLotteryNum

◦ Write an if, else-if, else statement, that will output to the screen the lottery number that is created at each iteration of the for loop. Make sure that the output states whether the random number is the first, next, or last lottery number. You can determine this by looking at the iterator of your for loop.

1. Compile and run your program, and make sure that you get similar output that is shown in Figure 1.

**III. Writing a Program with Methods**

For this part of the lab, a java program file without methods is provided for you on the schedule page of the

course website. The program reads two integers from the keyboard, and outputs a third integer, which is the first integer raised to the power indicated by the second integer. For example, if the first integer is 3, and the second integer is 4, then the program calculates and outputs 34 = 3 x 3 x 3 x 3 = 81. Two sample runs of the program are shown in Figure 2.

To get started, download and save to your computer, the file *MyProgramWithoutMethods.java*. Compile and run the program (Also shown in Figure 3).

This program calculates the nth power of a base integer input by the user. Please input your base; an integer less than 10 : 5

Please input your exponent; a positive integer less than 10 : 7

The integer 5 raised to the 7th power: 78125

This program calculates the nth power of a base integer input by the user. Please input your base; an integer less than 10 : 3

Please input your exponent; a positive integer less than 10 : 3

The integer 3 raised to the 3th power: 27

**Figure 2**: Two sample runs of the program MyProgramWithoutMethods.java

1 // import statements

2 import java.util.Scanner;

3

4 public class MyProgramWithoutMethods {

5

6 // the main routine

7 public static void main (String[] args) {

8

9 // print to the screen, instructions

10 System.out.println("This program calculates the nth power of a base integer input by the user.");

11 System.out.print("Please input your base; an integer less than 10 : ");

12

13 // receive user's input

14 Scanner keyboard = new Scanner(System.in);

15 int myInteger = keyboard.nextInt();

16 System.out.print("Please input your exponent; a positive integer less than 10 : ");

17 int myExponent = keyboard.nextInt();

18 int myIntegerPower = myInteger;

19

20 // perform calculation

21 for (int i = 2; i <= myExponent; i++){

22 myIntegerPower \*= myInteger;

23 }

24

25 // output result

26 System.out.println("The integer " + myInteger + " raised to the " + myExponent + "th power: " + myIntegerPower);

27 }

28 }

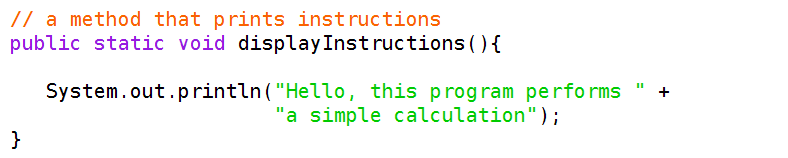
**Figure 3:** The complete MyProgramWithoutMethods.java file

At this point, you have a fully functioning java program. However, as mentioned in class, the program's main method is a bit cluttered, and it is not easy to read. In the subsequent steps, In this step, you'll modify your program by writing several methods. This will “un”clutter the main method.

**IIIa. Adding a printInstructions() Method**

The goal of this step is to simplify the main routine. To do that, you'll write a method that achieves the same functionality that is shown in lines 10-11 of your MyProgramWithoutMethods.java program in Figure 3.

Recall from lecture that the simplest type of method is one that does not return a value, nor receives a parameter. In class, we defined and used the following displayInstructions() method:



In this step of the lab, you'll write your own displayInstructions method. To do so:

1. Create a new java file. Cut and paste the code from your *MyProgramWithoutMethods.java* into the new just-created new java file.

2. Change the class name in the new java file to *MyProgramWithOneMethod*, and save the java file as

*MyProgramWithOneMethod.java*

3. In order to create a new method, you have to write the correct method's header, and body (refer to the slides from lecture if you need to review). Write a new method, that returns void, receives zero parameters, and is called printInstructions. The header of the method is the following, which should be written in your java file, above the main method:

public static void printInstructions()

4. The body of this new printInstructions method should contain a single println statement:

System.out.println("This program calculates the nth power of a base " + "integer input by the user.");

5. Edit the main routine so that instead of :

System.out.println("This program calculates the nth power of a base " + "integer input by the user.");

you invoke the printInstructions method:

printInstructions();

Recall from lecture that when invoking a method, you do NOT write the method modifiers that are part of the method's header; you also do not write the method's return type. You JUST write the method's name, and any input parameters, in parentheses, if the method requires parameters. In this first case, printInstructions() does not require any input parameters.

6. Compile and run your program. It should behave exactly as the first program. Except this time, the instructions are printed by the printInstructions method invoked from within the main method. Your code of your *MyProgramWithOneMethod.java* file should look similar to Figure 4.

1 *II* import statements

2 import java.uti1.Scanner;

3

4 public class MyProgramWithOneMethod {

5

6 *II* a method to print instructions

7 public static void printlnstructions{){

8 System.out.println{"This program calculates

9 of a base integer 1nput

10 }

11

12 a method to read an integer value from the

13 public static int getlntegerFromKeyboard{){

14 Scanner keyboard= new Scanner{System.in);

15 int keyboardlnput = keyboard.nextint{);

16 return keyboardlnput;

17 }

18

19 *II* the main routine

20 public static void main {String[] args){

21

22 *II* print to the screen, instructions

23 printInstructions();

24

25 *II* receive user's input

26 Scanner keyboard = new Scanner(System.in);

the nth power +

by the user. );

keyboard

27 System.out.print(''Please input your base; '' +

28 ''an integer less than 10 : '' ) ;

29 int mylnteger = keyboard.nextint();

30 System.out.print(''Please input your exponent; ''

31 + "a positive integer less than 10

32 int myExponent = keyboard.nextint();

33 int mylntegerPower = mylnteger;

34

35 *II* perform calculation

" ) ;

36 for {int i = 2; i <= myExponent;

37 mylntegerPower \*= mylnteger;

38 }

39

40 *I* output result

41 System.out.println( The integer

i++){

+ myinteger + ra1sed to the

42

43 }

44}

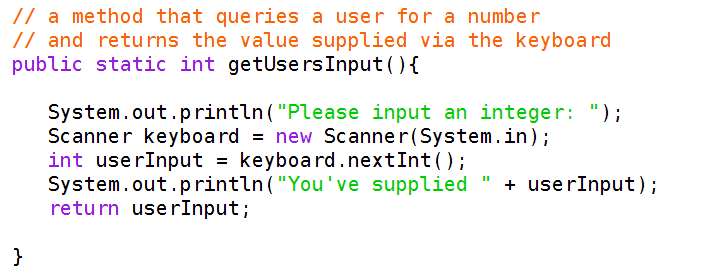
+ myExponent +

th power: + ffiyintegerPower);

Figure 4: The class MyProgramWithOneMethod, with the method print Instructions()

**IIIb. Adding a getIntegerFromKeyboard() Method**

In this section, you will write a method that instantiates the Scanner class, receives input from the user, and then returns the integer that was input via the keyboard. In lecture, we wrote the following method:



In this step of the lab, you'll add a getIntegerFromKeyboard method to your program, and you'll call that method twice, once to retrieve the base integer, and the second time to retrieve the exponent integer. The lesson here is that you can reuse a method as many times as needed. To complete this step:

1. Create a new java file, and cut and paste the code from your *MyProgramWithOneMethod.java* into the just-created new java file.

2. Change the class name in the new java file to MyProgramWithTwoMethods, and save the file as

*MyProgramWithTwoMethods.java*

3. Write a new method, that receives zero parameters, reads an integer input from the keyboard, and returns that value of type integer. Name your method getIntegerFromKeyboard. The method's header, and body, should be the following:

// a method that instantiates a Scanner class, listens to the keyboard,

// retrieves an integer, and returns the value public static int getIntegerFromKeyboard(){

Scanner keyboard = new Scanner(System.in); int keyboardInput = keyboard.nextInt(); return keyboardInput;

}

4. Modify the main method of your program, so that it retrieves the input from the keyboard, using the

getIntegerFromKeyboard method:

// receive user's input

System.out.println("Please input your base; an integer less than 10 : ");

int myInteger = getIntegerFromKeyboard(); System.out.println("Please input your exponent; a positive" +

" integer less than 10 : ");

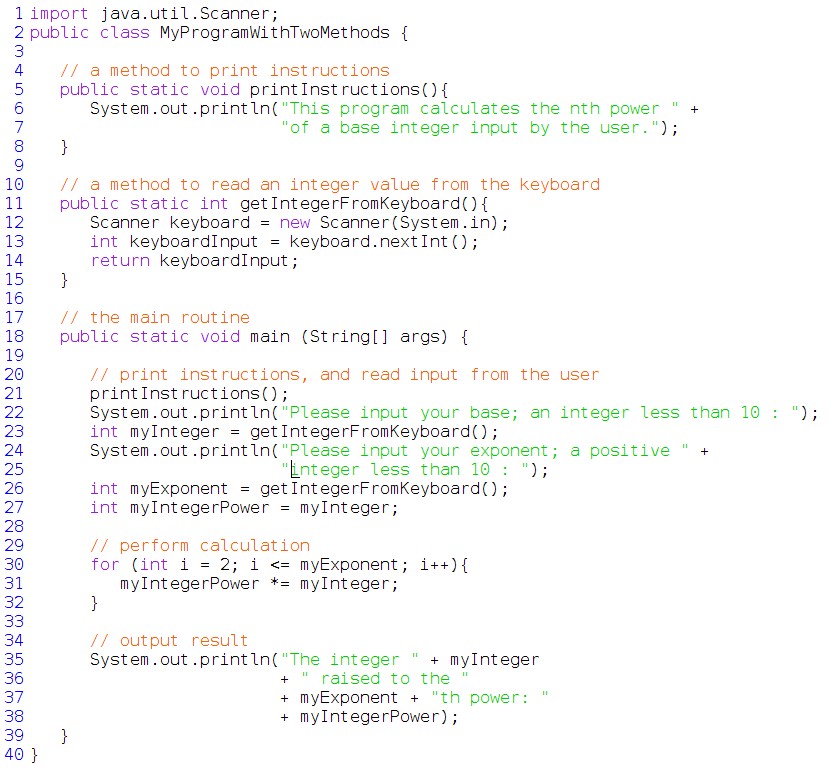
int myExponent = getIntegerFromKeyboard();

int myIntegerPower = myInteger;

Be sure to remove the statement Scanner keyboard = new Scanner(System.in); from your main method. Notice how you are using the getIntegerFromKeyboard method twice.

5. Compile and debug your program, and make sure that it performs the same way as your previous two

programs in this part of the lab. Your program should look similar to what is shown in Figure



**Figure 5:** The MyProgramWithTwoMethods program, that invokes methods at lines 21, 23, and 26

**IIIc. Adding the calcExponentValue Method to your program**

In this final section, you'll write a method that takes two arguments – both of them integers – and calculates the value of raising the first integer to the power indicated by the second integer. For example, if the first integer provided by the user is 3, and the second integer provided by the user is 4, then the method would calculate

34 = 3 x 3 x 3 x 3 = 81. To complete this step:

1. Create a new java file. Cut and paste the code from your *MyProgramWithoutTwoMethods.java* into the just-created new java file.

2. Change the class name in the new java file to MyProgramWithThreeMethods, and save the file as

*MyProgramWithThreeMethods.java*

3. Write the following method:

// a method that takes two arguments, and calculates the first input

// argument raised to the power of the second argument

public static int calcExponentValue(int integerValue, int exponentValue){

int integerPower = integerValue;

for (int i = 2; i <= exponentValue; i++){

integerPower \*= integerValue;

}

return integerPower;

}

Notice that the method calcExponentValue takes two arguments, the first an integer, and the second also an integer. When you invoke the method from the main routine of your program, you have to pass the values in the

correct order.

4. Invoke the calcExponentValue method in the main part of your program.

5. The final, main routine is the following, where the calcExponentValue method is called directly within the System.out.println statement.

// the main routine

public static void main (String[] args) {

printInstructions();

System.out.print("Please input an integer less than 5: ");

int myInteger = getIntegerFromKeyboard();

System.out.print("Please input a positive, integer exponent < 5: ");

int myExponent = getIntegerFromKeyboard(); System.out.print("The integer " + myInteger);

System.out.println(" raised to the " + myExponent + "th power is: "

+ calcExponentValue(myInteger,myExponent));

}

6. After you've added the calcExponentValue method into your class, and after you've invoked that method in your main method, compile and run your program, which should look similar to Figure 6. Test it on a few sample inputs. If it works as expected, then you are done. Notice how the main method, which now

**includes invocations of several other methods, is much easier to read than the original main method of the**

MyProgramWithoutMethods **class.**

1 import

2 public

3

java.util.Scanner;

class MyProgramWithThreeMethods {

4 *II* a method to print instructions

5 public static void print!nstructions(){

6 System.out.println(''This program calculates

7 ''of a base integer input

8 }

9

10 *II* a method to read an integer value from the

11 public static int getintegerFromKeyboard(){

12 Scanner keyboard = new Scanner(Systerr.in);

13 int keyboard!lput = keyboard.next!nt();

14 return keyboard!nput;

15 }

16

the nth power " by the user.");

keyboard

17 *II* a method that takes two arguments, and performs a calculation

18 public static int calcExponentValue(int integerValue, int exponentValue){

19 int integerPmver = integerVa1ue;

20 for (int i = 2; i <= exponentValue; il {

21 integerPower \*= integerValue;

22 }

23 return integerPower;

24 }

25

26 *II* the main routine

27 public static void main (String[] args){

28

29 *II* print instructions, and read input from the user

30 print!nstructions();

31

32

33

34

35

36

37

38

39

40

41

42

43 }

System.out.print("P1ease input your base; an integer less than 10 int my!nteger = getintegerFromKeyboard();

System.out.print "Please input your exponent; a positive " +

"integer less than 10 : ");

int myExponent = getintegerFromKeyboard();

*II* output resJlt

System.out.println(''The integer '' + rry!nteger

+ '' raised to the ''

+ myExponent + "th power: "

+ calcExponentValue(myinteger, myExponent));

}

II ) ***i***

**Figure6: Complete final program, with three methods (lines 5-8,11-15, and 18-24)**

**which are invoked in lines 30, 32,** 35, **and 41.**

**IV. What to hand in**

The following files should be uploaded to Canvas (use the “Add Another File” option to submit multiple files):

*“TerraceEverVerdant.java”*

*“TerraceEverVerdant Output.docx”*

*“GenerateRandomLotteryNums.java”*

*“GenerateRandomLotteryNums.java Output.docx”*

*“MyProgramWithOneMethod.java “MyProgramWithOneMethod Output.docx”*

*“MyProgramWithTwoMethods.java” “MyProgramWithTwoMethods Output.docx”*

*“MyProgramWithThreeMethods.java”*

*“MyProgramWithThreeMethods Output.docx”*

Be sure that each .java file is commented, and be sure to include your name at the top of each file. Code must be indented, so that it is easy to read. Finally, make sure that you have given your variables good descriptive names.

**V. Rubric**

|  |  |
| --- | --- |
| **File / task** | **Points** |
| *TerraceEverVerdant.java* compiles and generates correct ASCII image ( in output file ) | 30 |
| *GenerateRandomLotteryNums.java* compiles and generates lottery numbers(in output file ) | 30 |
| *MyProgramWithOneMethod.java* compiles and generates correct output ( in output file ) | 10 |
| *MyProgramWithTwosMethods.java* compiles and generates correct output ( in output file ) | 10 |
| *MyProgramWithThreeMethods.java* compiles and generates correct output ( in output file ) | 10 |
| All .java and text files are commented, and contain your name and date | 5 |
| Variable names are adequate and descriptive in all java files | 5 |
| Total | 100 |