# Practical Assessment

This assignment relates to the following Course Learning Requirements:

CLR 1:  Define, analyze, and document the logic of a solution to a given problem.

CLR 2: Implement the solution to a given problem by writing the appropriate code in a high-level language (Java).

CLR 4: Install and use the Java Development, Runtime Environment, and documentation libraries.

CLR 7: Create documentation and a Java solution for programming problems that adhere to the submission standard identified within the timeframe given in the problem description.

Objective of this Assignment:

The objective of this assignment is to design a game between a player and computer as mentioned below:

The rules of the game are simple: the game starts with 4 types of files – PDF, DOCX, XLSX, and JAVA. There will be a random number of PDF files, between 3 and 7. From this number, there will be 2 more DOCX files, 4 more XLSX files, and 6 more JAVA files. For example, if there are 3 PDF files, there would be 5 DOCX, 7 XLSX, and 9 JAVA files. The players, beginning with your user, will take turns removing a number of files from directories. In a single turn, a player must select a file type, and choose how many files they wish to remove. The player may take no more than half of the files in a directory, unless it is the final file in the directory. They must remove at least one file in each turn. The player to remove the very last file loses.

# Pre-Assignment Instructions:

1. To prepare you for this assignment, read the module 3 content and follow the embedded learning activities.
2. Access this assignment in Brightspace. You may complete the assignment locally on your machine in a plain text editor.

**Assignment Steps:**

1. Create a Java class named **File**.
2. **File** should have a **private** instance variable of type **int** named **numFiles**.
3. Create a parameterized (aka initial) constructor with a single parameter of type **int**, which initializes the **numFiles** property of the class.
4. Write a **getter** (aka accessor) method for the **numFiles** property. Make sure you use the proper naming convention and return type for getter methods.
5. Write a **boolean** method named **isEmpty** that will return **true** if **numFiles** is equal to 0, otherwise it will return **false**.
6. Write a **boolean** method named **removeFiles** with a single parameter of type **int** – the number of files the player is attempting to remove from the directory. Conditions and messages to display:
7. If the user is attempting to remove less than 1 file, output the following error message and return **false**.

Sorry, you must remove at least one file!

1. If the user is attempting to remove more than half of the files in the directory and there are more than 1 file in the directory, output the following error message and return false.

Sorry, that's too many files!

1. Otherwise, reduce the number of files in the directory by the requested number of files specified in the parameter and return **true**.
2. Create a Java class named **Game**.
3. The **Game** class will use the **Scanner** and **Random** classes of the **java.util** package as building blocks.
4. Game should have the following **private** properties declared:
5. **String playerName**
6. **int player**
7. **Scanner input**
8. **Random rand**
9. The following 4 objects of **File** class named: **pdf**, **docx**, **xlsx**, **java**.
10. Create a no-args (explicit default) public constructor.
11. Initialize the **input** variable with a new **Scanner**, using the default input stream, **System.in**.
12. Initialize the **rand**variable with a new **Random**object, using the default constructor of the **Random** class.
13. Initialize the player variable with the value 1 (indicating it is the turn of player 1).
14. Initialize the **playerName** instance variable:
15. Print a prompt to the end user as shown below, to let them know your program is expecting some input.

Please enter your name:

1. Using the newly initialized **input** variable and the **next** method of the **Scanner** class, initialize the **playerName** instance variable.
2. Using the newly initialized **rand** variable and the **nextInt** method of the **Random** class, initialize a local **int** variable with a random value between 3 and 7 (inclusive).
3. If the random value generated is evenly divisible by 2, add 1 to the starting value to make it an odd number. The resulting value should be either 5 or 7.
4. Initialize the 4 **File** variables:
5. Initialize the **pdf** **File** variable with the random starting value selected in the previous step as an argument to its constructor (3,5 or 7).
6. Initialize the **docx File** variable with an argument 2 more than the random starting value (5, 7 or 9).
7. Initialize the **xlsx File** variable with an argument 4 more than the random starting value (7, 9 or 11).
8. Initialize the **java File** variable with an argument 6 more than the random starting value (9, 11, or 13).
9. Print the following welcome statement to standard output:

Welcome to the Game

You'll need to first pick a file type

Then select how many files you wish to remove

1. Write a **void** helper method called **displayGame** that will print each file type and the number of files remaining. Use the **getNumFiles** method of the **File** class with each **File** object (**pdf**, **docx**, **xlsx**, **java**) to get the number to display.
2. Eg:

PDF DOCX XLSX JAVA

 3   5    7    9

1. Write a **void** method called **playGame** that will execute each turn of the game.
2. Begin by invoking (aka calling or executing) the **displayGame** method.
3. Declare a local variable(object) of type **File** named **selectedFile** and initialize it to the value **null**.
4. Declare a local variable of type **int** named **numFilesToRemove** and initialize it with the value 0 (zero).
5. If **player** equals 1 (this is the user’s turn):
6. **Do** the following **while** the **selectedFile**’s **isEmpty** method returns **True** (do not let the user select a File type that has no files in its directory anymore):
7. Display a prompt to the end user, using the **playerName**, to select a file type, as shown below:

<playerName>, select a file type:

Press 1 for PDF

Press 2 for DOCX

Press 3 for XLSX

Press 4 for JAVA

1. Using the**nextInt** method of the **input** object, get the end user’s response. You may assume they will enter an int value from the keypad on their phone.
2. Use a **switch**control statement to process their selection.
3. If they enter 1, initialize the **selectedFile** variable to the **pdf** **File** object.
4. If they enter 2, initialize the **selectedFile** variable to the **docx** **File** object.
5. If they enter 3, initialize the **selectedFile** variable to the **xlsx** **File** object.
6. If they enter 4, initialize the **selectedFile** variable to the **java File** object.
7. Don’t forget to **break** out of each case!
8. Add a **default** **case** to handle any unexpected user input!
9. Display a prompt to the end user to enter the number of files they wish to remove:

Enter number of files to remove:

1. Use the **nextInt** method of the**input** object to initialize the **numFilesToRemove** variable with the user’s response.
2. You may assume the user will enter an integer value(**int**).
3. Otherwise, if player equals 2 (this is the computer’s turn):
4. Display the following to simulate a prompt to the computer:

Computer, select a file type:

1. Do the following while the selectedFile’s **isEmpty** method returns **True**:
2. Declare a local **int**variable named **randFileType**, and initialize it using the **nextInt** method of the **rand**object to a value between 1 and 4, inclusive.
3. Use a **switch** control statement to process **randFileType**.
4. If **randFileType** equals **1**:
5. Set **selectedFile** to the **pdf** object.
6. If **selectedFile isEmpty**, continue to jump to the loop condition.
7. Otherwise, display the selection “PDF”.
8. Create similar **cases** in the **switch** for values **2**, **3**, and **4** with **docx**, **xlsx**, and **java** file types.
9. Don’t forget your **break** statements!
10. Display the following to simulate a prompt to the computer:

Computer, choose number of files to remove:

1. If the**selectedFile getNumFiles** method returns a value of **1**, initialize the **numFilesToRemove** variable to **1**.
2. Otherwise, initialize the numFilesToRemove variable with a random number of files between 1 and half the number of files of the file type.
3. numFilesToRemove = rand.nextInt(selectedFile.getNumFiles() / 2) + 1;
4. Now that the file type and number of files have been chosen, attempt to remove the files from the directory.
5. Declare a local **boolean** variable named **itWorked**.
6. Assign the return value of executing the **removeFiles** method of the **selectedFile** with the **numFilesToRemove** as an argument.
7. boolean itWorked = selectedPile.removeFiles(numFilesToRemove);
8. If the move did NOT work, execute the **return** statement to return to the **main** method.
9. Next, swap players. In programming terms, we call this **toggling**a value.
10. If **player** equals 1, assign **player** the value of 2. Otherwise (**player** equals 2), assign **player** the value of 1.
11. Declare a method of return type **int** named **determineWinner**.
12. Declare a local variable of type **int** named **winner**, with an initial value of zero (0).
13. Declare a local variable of type **int** named **allFiles**, with a value equal to the sum of calling **getNumFiles** of every **File** object in the game.
14. int allFiles = pdf.getNumFiles() + docx.getNumFiles() + …
15. If **allFiles** equals zero (0), the game is over. Assign the value of the **player**variable to the **winner** variable.
16. If **winner** equals 1, print the following to standard output:

<playerName> Wins!

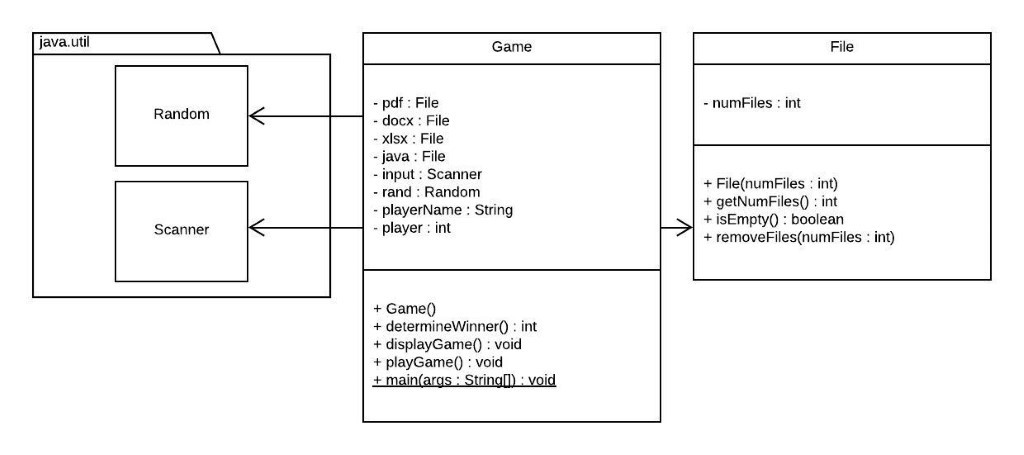
1. Otherwise, print this:

Computer Wins!

1. **Return** the value of **winner**.
2. Declare the main method of the program:
3. The declaration is always the same: public static void main(String[] args)
4. Declare a **new Game** object named **game**, using the **Game**class’s default constructor.
5. **Do** the following **while** the **determineWinner** method of the **game**object equals zero (0):
6. Execute the **playGame** method of the **game** object.
7. Output the following message:

Thanks for playing!

1. UML Class Diagram for reference



1. Your assignment is to be submitted on Brightspace as Java files. It should be submitted with the following guidelines:
2. Include the file header using the template provided in every java file submitted.
3. Follow expected style guidelines:
4. Use “Egyptian” style braces for all classes and methods.
5. Indent your code using 4 spaces (no tabs).
6. Follow naming conventions for all class, variable, and method identifiers.
7. Use appropriate whitespace for readability.
8. Comment your code (no less than one comment per class, and one per method).
9. Name the files as  **File.java** and **Game.java**.
10. Feel free to make any assumption you need to make to implement a working code. Make sure the code delivers expected successful output. Highlight any assumption you may make while submitting the assessment.