**Virtual Key for Your Repositories**

Project 1

DESCRIPTION

**Project objective:**

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**Background of the problem statement:**

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You’re asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you’re asked to present the following in the next 15 working days (3 weeks):

* Specification document - Product’s capabilities, appearance, and user interactions
* Number and duration of sprints required
* Setting up Git and GitHub account to store and track your enhancements of the prototype
* Java concepts being used in the project
* Data Structures where sorting and searching techniques are used.
* Generic features and three operations:
  + Retrieving the file names in an ascending order
  + Business-level operations:
    - Option to add a user specified file to the application
    - Option to delete a user specified file from the application
    - Option to search a user specified file from the application
    - Navigation option to close the current execution context and return to the main context
  + Option to close the application

The goal of the company is to deliver a high-end quality product as early as possible. 

**The flow and features of the application:**

* Plan more than two sprints to complete the application
* Document the flow of the application and prepare a flow chart
* List the core concepts and algorithms being used to complete this application
* Code to display the welcome screen. It should display:
  + Application name and the developer details
  + The details of the user interface such as options displaying the user interaction information
  + Features to accept the user input to select one of the options listed
* The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it
* The second option should return the details of the user interface such as options displaying the following:
  + Add a file to the existing directory list
    - You can ignore the case sensitivity of the file names
  + Delete a user specified file from the existing directory list
    - You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
    - Return a message if FNF (File not found)
  + Search a user specified file from the main directory
    - You can add the case sensitivity on the file name to retrieve the correct file
    - Display the result upon successful operation
    - Display the result upon unsuccessful operation
  + Option to navigate back to the main context
* There should be a third option to close the application
* Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

**You must use the following:**

* Eclipse/IntelliJ: An IDE to code for the application
* Java: A programming language to develop the prototype
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Search and Sort techniques: Data structures used for the project
* Specification document: Any open-source document or Google Docs

**Following requirements should be met:**

* The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
* Document the step-by-step process starting from sprint planning to the product release.
* Application should not close, exit, or throw an exception if the user specifies an invalid input.
* You need to submit the final specification document which includes:
  + Project and developer details
  + Sprints planned and the tasks achieved in them
  + Algorithms and flowcharts of the application
  + Core concepts used in the project
  + Links to the GitHub repository to verify the project completion
  + Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

==============================================================================================

Answer:

**Get the team to know each other. And have a scrum meeting:**

**Define the backlogs:**

PO will identify the items included in this project.

Development team will estimate the efforts and time needed to achieve.

Develop a "user story" to agree between product owner and development team about which item need to be working on.

Then **make the users story** in backlog item format like:

Retrieving the file names in an ascending order

I as a user need to have an application the has an Option to add a user specified file to the application

Option to delete a user specified file from the application

Option to search a user specified file from the application

Navigation option to close the current execution context and return to the main context

Option to close the application.

Write the user stories on **indexed cards to and hanged on a board or wall**.

User story cards need to have item identifier name and description , estimated value from product owner perspective estimated efforts from development team

**Apply:**

**INVEST**

Time boxing:

This sprint will be every 2 -4 week to plan each sprint, since we will need two sprints then we will need 4 to 8 weeks to accomplish the project.

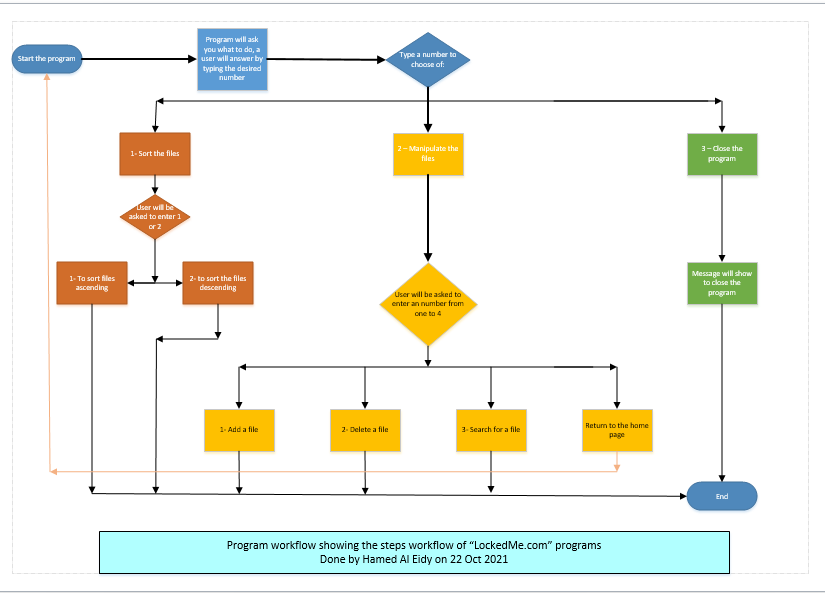
Also, We will need a sprint planning meeting ( 2 hours a week).

Daily Scrums: 15 minutes.

Sprint review: 1 hour every week.

-----

**Project step by step workflow:**



Source code:

==================================================================================================

/\*\*

\* Program as an answer to OOPS assissment

\* Allow user to select from a pre defined list & how to sort filenames in ascending or descending order

\* All Rights Reserved

\* Version 1.0

\* 10/21/2012

\* Done by Hamed Al Eidy

\*/

**import** java.util.Scanner;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** Assis {

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

*master*();

String[] arr = {"1. Sort the files names in ascenting order.",

"2. Manipulate files",

"3. Close the Application"};

Scanner input = **new** Scanner(System.***in***);

System.***out***.println("Type the service you want to do: ");

**int** n = input.nextInt();

**if** (n > arr.length || n < arr.length - arr.length+1) {

System.***out***.println("Invalide selection, please select 1, 2 or 3");

}

**switch** (n) {

**case** 1: {

System.***out***.println("1. Sorting file names in ascending order?\n"

+ "2. Sorting file names in descending order?\n");

*Sortfiles*();

**break**;

}

**case** 2: {

System.***out***.println("Do you want to:\n"

+ "1. Add a file.\n"

+ "2. Delete a file.\n"

+ "3. Search for a file\n"

+ "4. Go to previous selections.\n"

+ "Please type which action.\n");

*manipSelection*();

**break**;

}

**case** 3: {

*closeApp*();

**break**;

}

}

}

**private** **static** **void** Sortfiles() {

String[] sort= {"1. Asecendingly","2. Desendingly"};

Scanner s1 = **new** Scanner(System.***in***);

**int** ss = s1.nextInt();

**if** ( ss > 0 && ss < 3) {

**switch** (ss) {

**case** 1:{

File fileDir = **new** File("C:\\Users\\212397697\\Box Sync\\Programming\\Assis");

List listFile = Arrays.*asList*(fileDir.list());

System.***out***.println("Listing files unsorted");

Collections.*sort*(listFile);

System.***out***.println("---------------------------------------");

System.***out***.println("Sorting by filename in ascending order");

} **break**;

**case** 2:{

File fileDir2 = **new** File("C:\\Users\\212397697\\Box Sync\\Programming\\Assis");

List listFile2 = Arrays.*asList*(fileDir2.list());

System.***out***.println("Listing files unsorted");

System.***out***.println("---------------------------------------");

System.***out***.println("Sorting by filename in descending order");

Collections.*sort*(listFile2,Collections.*reverseOrder*());

**break**;

}

}

} **else** {

System.***out***.println("Invalid Entery, please select 1 or 2.");

}

}

**private** **static** **void** manipSelection() {

String[] maniparray={"1. Add a file." ,

"2. Delete a file.",

"3. Search for a file",

"4. Go to previous selections."};

Scanner m1 = **new** Scanner(System.***in***);

**int** m = m1.nextInt();

**if** (m < 5 && m > 0) {

**switch** (m) {

**case** 1:{

File file = **new** File("C:\\Users\\212397697\\Box Sync\\Programming\\Assis\\new text.txt");

**boolean** result;

**try** {

result = file.createNewFile(); //creates a new file

**if**(result) { // test if successfully created a new file

System.***out***.println("file created "+file.getCanonicalPath()); //returns the path string

} **else** {

System.***out***.println("File already exist at location: "+file.getCanonicalPath());

}

}

**catch** (IOException e)

{

e.printStackTrace(); //prints exception if any

} **break**;

}

**case** 2:{

File f= **new** File("C:\\Users\\212397697\\Box Sync\\Programming\\Assis\\new text.txt"); //file to be delete

**if**(f.delete()) { //returns Boolean value

System.***out***.println(f.getName() + " deleted"); //getting and printing the file name

} **else** {

System.***out***.println("failed");

} **break**;

}

**case** 3:{

File dir = **new** File("C:\\Users\\212397697\\Box Sync\\Programming\\Assis");

String[] children = dir.list();

**if** (children == **null**) {

System.***out***.println("does not exist or is not a directory");

} **else** {

**for** (**int** i = 0; i < children.length; i++) {

String filename = children[i];

System.***out***.println(filename);

}}

**break**;

}

**case** 4:{

*master*();

**break**;

}}

} **else**

System.***out***.println("Invalid entery");

}

**private** **static** **void** closeApp() {

System.***out***.println("Closing your application... \nThank you!");

}

**public** **static** **void** master() {

System.***out***.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.***out***.println("\tWelcome to LockedMe.com \n");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("\nKindly select the service you want to do:\n"

+ "1. Sort the files names in ascenting order.\n"

+ "2. Manipulate files.\n"

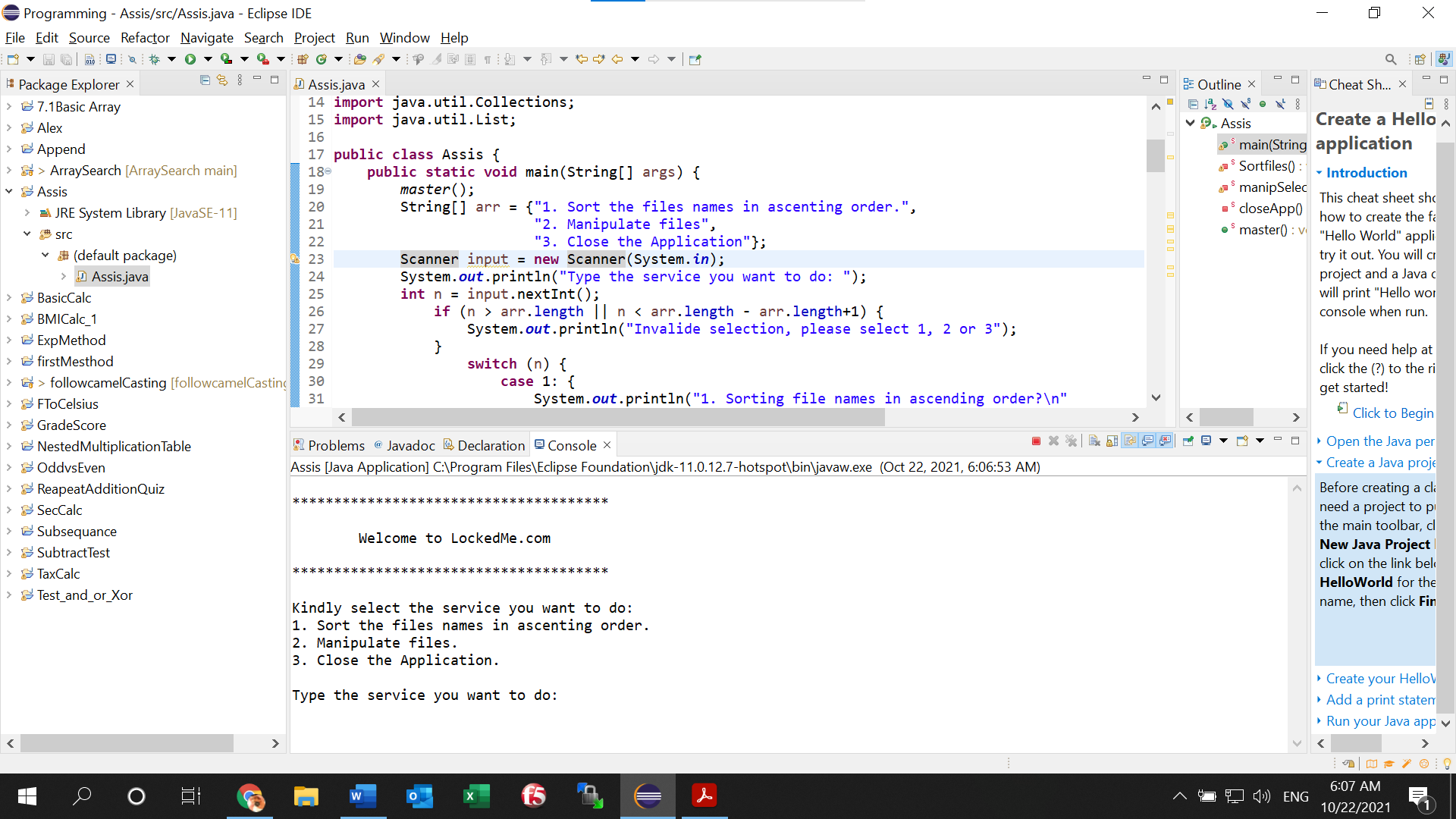
+ "3. Close the Application.\n");

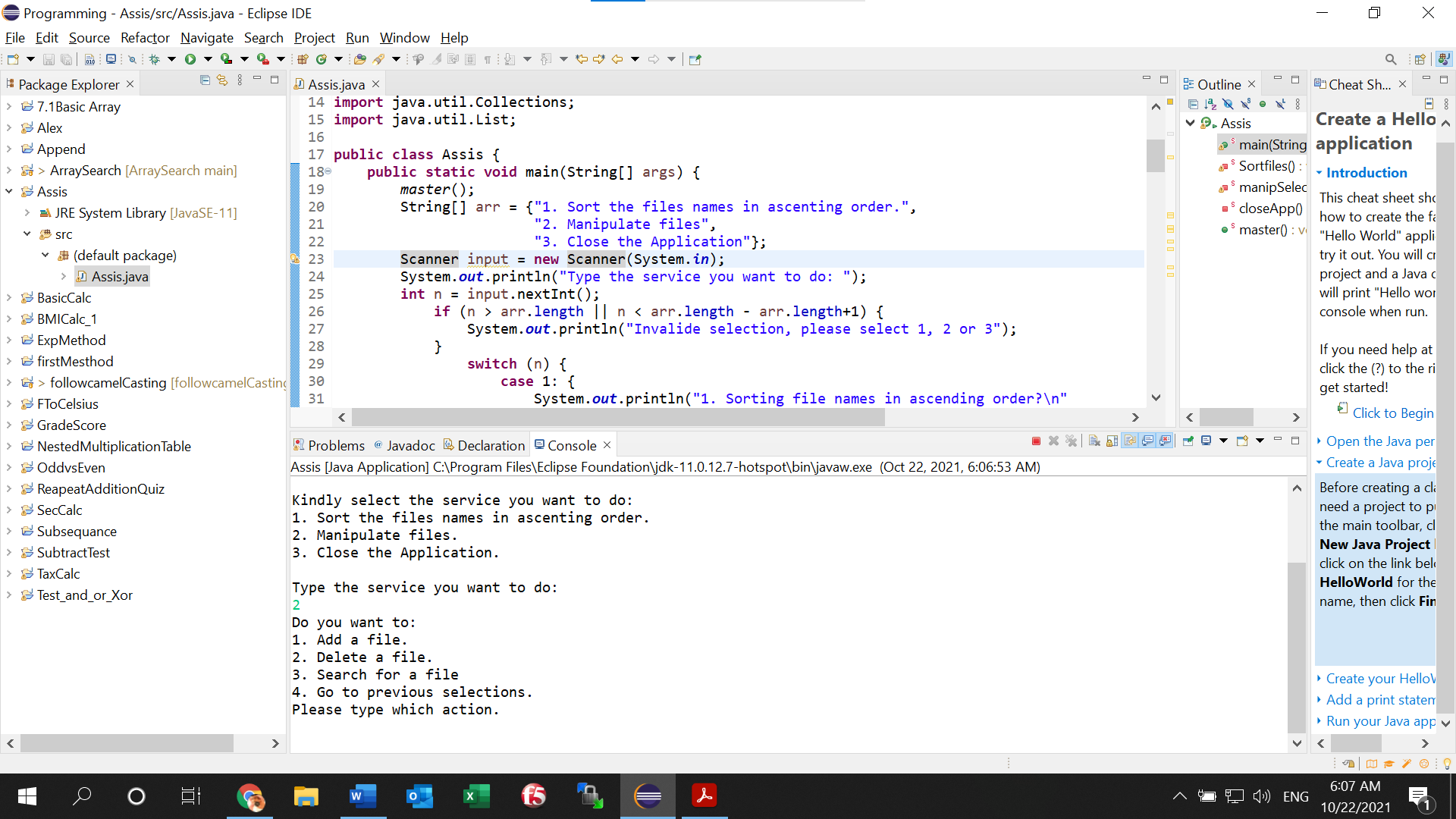
}

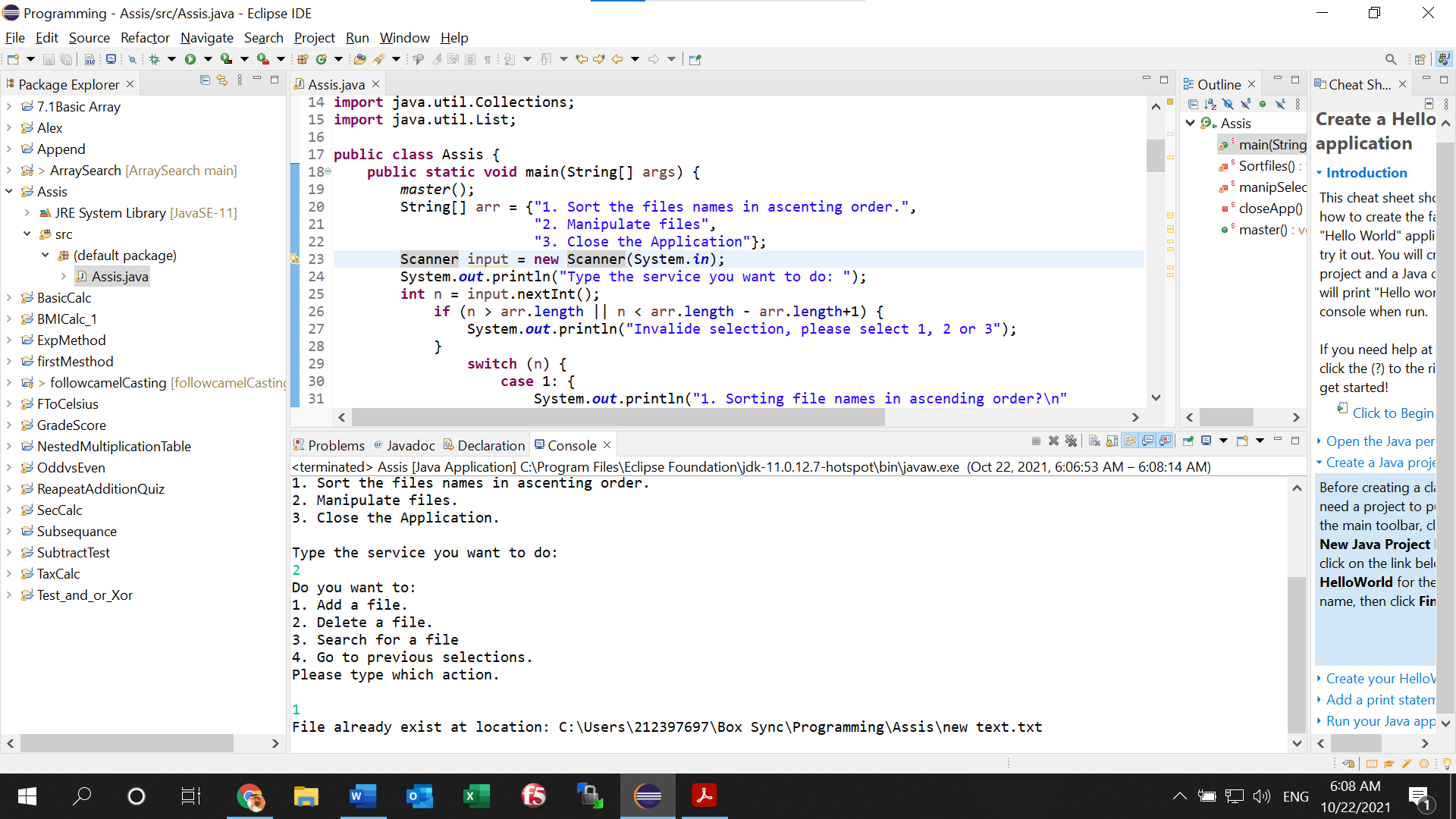
}

================================================================================================

Screenshots:







=== End ===