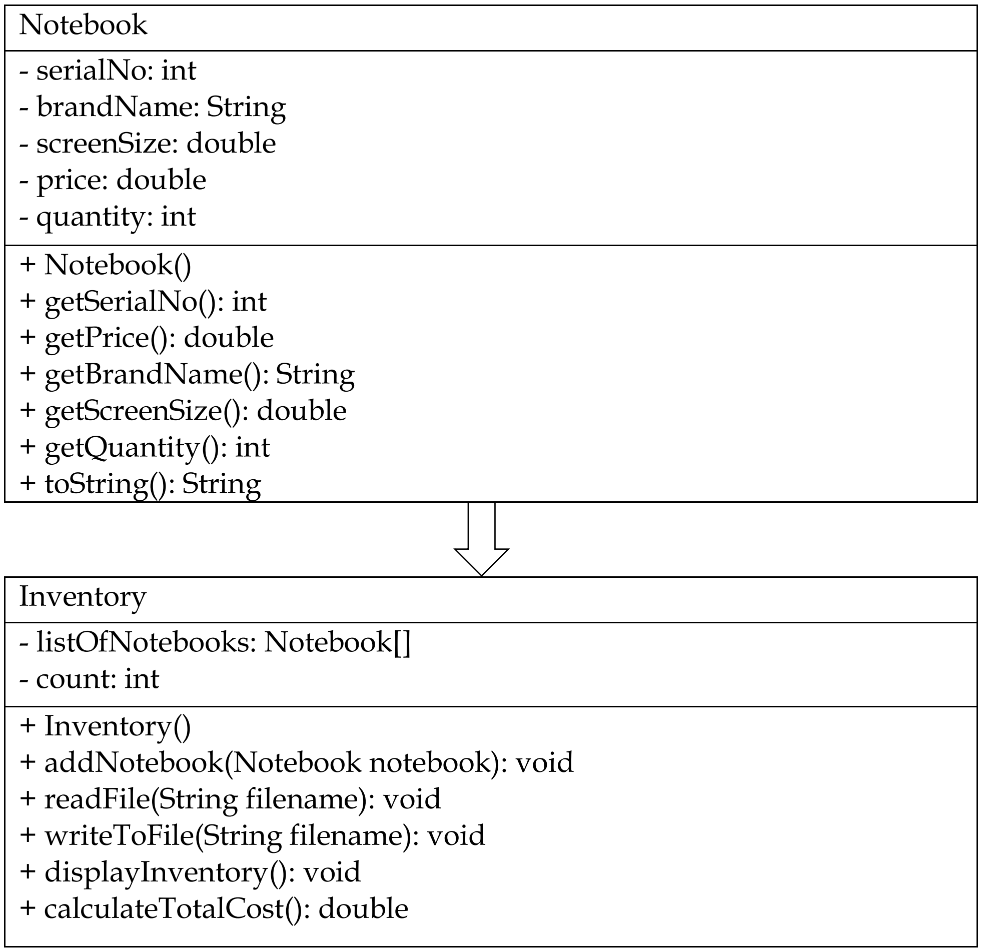
Entrance Exam – Part 2 (39 points)

Due: Friday, May 20 @ 11:55pm

Note: You will find at the end of this document some functions that you may find useful when answering the questions of this exam.

Programming Questions

Use the UML class diagrams for the Notebook and Inventory classes and the method descriptions. You will be writing the code for the constructor and methods in the Inventory class in the following questions 1 - 6.



Notebook
- serialNo: int - brandName: String - screenSize: double - price: double - quantity: int
+ Notebook() + getSerialNo(): int + getPrice(): double + getBrandName(): String + getScreenSize(): double + getQuantity(): int + toString(): String

Inventory
- listOfNotebooks: Notebook[] - count: int
+ Inventory() + addNotebook(Notebook notebook): void + readFile(String filename): void + writeToFile(String filename): void + displayInventory(): void + calculateTotalCost(): double

1. **(6 points)** Declare the instance variables and write the Default (no-arg) constructor. Create the listOfNotebooks array in the default constructor. There will be no more than 50 Notebook instances that will be stored in the array.

2. **(6 points)** Write the method `addNotebook()`. This method will add a new Notebook object to the listOfNotebooks. If the array is full, display a message "No more notebooks can be added."

3. **(6 points)**

Write the method `readFile()`. This method will accept the name of the file to read as a parameter for an unknown number of Notebook data from a file called Notebooks.txt into the array listOfNotebooks in the same class and handle any exceptions. Each item in the file has the following format, one immediately after the other.

```
<serialNo>
<brandName>
<screenSize> <price> <quantity>
```

Write the method `writetoFile()`. This method will accept as a parameter the name of the file to write to. It will write the contents of the array(which is stored in this class) to the output file UpdatedNotebooks.txt. Remember to handle possible exceptions.

4. **(7 points)** Write the method `displayInventory()` that displays all the information about all the items in the listOfNotebooks . Use the toString() method of the Notebook class to format the output. The toString() method of the Notebook class will be used to format and write Notebook data into the output file. Do not write the toString() method.

5. **(7 points)** Write the method `calculateTotalCost()` that calculates the total inventory value of all of the items on the ListOfNotebooks.

6. **(7 points)** Write a static method `searchNotebookPrice()` that accepts two parameters, a Notebook array and a notebook serialNo. The method will search the Notebook array for the notebook with the specified serialNo and return the notebook's price. If the notebook is not found, return -1.0.

Supplement Java Functions:

Input and Output API

FileWriter- throws IOException
public FileWriter(String fileName) Constructs a FileWriter object given a file name
public FileWriter(File file) Constructs a FileWriter object given a File object.
public FileWriter(String filename, Boolean append) Constructs a FileWriter object given a file name with a boolean indicating whether or not to append the data written.
public FileWriter(File file, Boolean append) Constructs a FileWriter object given a File object.
PrintWriter- throws FileNotFoundException
public PrintWriter(Writer file) Constructs a PrintWriter object given a Writer object
File- throws NullPointerException
public File(String pathname) Creates a new File Instance by converting the given pathname string into an abstract pathname.
Scanner- throws FileNotFoundException
public Scanner(File source) Constructs a new Scanner that produces values scanned from the specified file.
public Scanner(String source) Constructs a new Scanner that produces values scanned from the specified string.

Scanner class interface – partial Method Summary

void	close() Closes this scanner.
String	next() Finds and returns the next complete token from this scanner.
boolean	nextBoolean() Scans the next token of the input into a boolean value and returns that value.
double	nextDouble() Scans the next token of the input as a double.
int	nextInt() Scans the next token of the input as an int.
String	nextLine() Advances this scanner past the current line and returns the input that was skipped.

boolean	hasNext() Returns true if this scanner has another token in its input
boolean	hasNextBoolean() Returns true if the next token in this scanner's input can be interpreted as a boolean value using a case insensitive pattern created from the string "true|false".
boolean	hasNextDouble() Returns true if the net token in this scanner’s input can be interpreted as a double value using the nextDouble() method
boolean	hasNextInt() Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the nextInt() method.
boolean	hasNextLine() Returns true if there is another line in the input of this scanner.