|  |  |
| --- | --- |
| NAME : | Muhammad Alif Serbaini |
| ID : | AM2111010447 |
| SECTION NO : | 01 |
| SUBJECT CODE : | SWC2333 |



**LAB TEST**

|  |  |  |
| --- | --- | --- |
| **COURSE** | **:** | **OBJECT ORIENTED PROGRAMMING** |
| **COURSE CODE** | **:** | **SWC2333/SWC2053** |
| **DURATION** | **:** | **2 HOURS** |

1. This lab test aims to fulfil CLO2 - Construct executable programs based on class diagram. (C3, PLO3)

1. The marks allocated for this lab test is 15% of total marks.

1. This lab test consists of **ONE (1)** question.

1. Answer ALL questions and it must be using Eclipse and Microsoft Words.

1. Write down your name, ID, Subject Code and Section No on the front page of your answer sheet.

1. After you have finished, save you answer including the rubric in one pdf format with your Name, Section No, the name of the assessment and the subject code as the file’s name. (*Example of the file’s name: Ahmad Albab S1 LabTest SWC2333*)

1. You must submit your answer ONLINE through Google Classroom by 16 June 2022 before 5:30PM

|  |
| --- |
| **DO NOT OPEN THIS QUESTION PAPER UNTIL YOU ARE TOLD TO DO SO** |

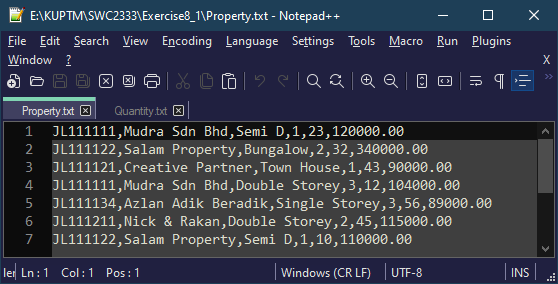
*This question paper consists of 3 printed pages including the front page.*

Coding :

*import* java.io.\*;  
*import* java.text.DecimalFormat;  
*import* java.util.StringTokenizer;  
  
*public class* PropertyData {  
 *public static void* main(String[] args) *throws* IOException {  
 DecimalFormat df = *new* DecimalFormat("#0.0");  
 *try* {  
 String Business\_registration, Property\_name, House\_Type;  
 *double* development\_phase, costperunit;  
 *int* total\_house;  
  
 BufferedReader br = *new* BufferedReader(*new* FileReader("Property.txt"));  
  
 FileWriter fw1 = *new* FileWriter("Quantity.txt");  
 PrintWriter pw1 = *new* PrintWriter(fw1);  
 FileWriter fw2 = *new* FileWriter("total.txt");  
 PrintWriter pw2 = *new* PrintWriter(fw2);  
  
 String line = br.readLine();  
 *//Looping to get Quantity of the houses  
 int* SemiD\_Quantity = 0, Bungalow\_Quantity = 0,TownHouse\_Quantity = 0, DoubleStorey\_Quantity = 0, SingleStorey\_Quantity = 0;  
 *//Looping to get the total cost of the houses  
 double* mudra = 0, salam = 0, creative = 0, azlan = 0, nick = 0;  
 pw2.println("The total cost for each Property is : ");  
 *while* (line != *null*) {  
 StringTokenizer st = *new* StringTokenizer(line, ",");  
 st.nextToken(); *//Skip Business Registration* Property\_name = st.nextToken();  
 House\_Type = st.nextToken();  
 st.nextToken(); *//Skip Development Phase* total\_house = Integer.*parseInt*(st.nextToken());  
 costperunit = Double.*parseDouble*(st.nextToken());  
  
 *//IF else to differentiate and detect the house type, and calculate the quantity of said house type.  
 if* (House\_Type.equals("Semi D")) {  
 SemiD\_Quantity = SemiD\_Quantity + total\_house;  
 } *else if* (House\_Type.equals("Bungalow")) {  
 Bungalow\_Quantity = Bungalow\_Quantity + total\_house;  
 } *else if* (House\_Type.equals("Town House")) {  
 TownHouse\_Quantity = TownHouse\_Quantity + total\_house;  
 } *else if* (House\_Type.equals("Double Storey")) {  
 DoubleStorey\_Quantity = DoubleStorey\_Quantity + total\_house;  
 } *else if* (House\_Type.equals("Single Storey")) {  
 SingleStorey\_Quantity = SingleStorey\_Quantity + total\_house;  
 }  
  
 *//if else to detect different property name and calculate the total cost for the property type  
 if* (Property\_name.equals("Mudra Sdn Bhd")) {  
 mudra = mudra + costperunit;  
 } *else if* (Property\_name.equals("Salam Property")) {  
 salam = salam + costperunit;  
 } *else if* (Property\_name.equals("Creative Partner")) {  
 creative = creative + costperunit;  
 } *else if* (Property\_name.equals("Azlan Adik Beradik")) {  
 azlan = azlan + costperunit;  
 } *else if* (Property\_name.equals("Nick & Rakan")) {  
 nick = nick + costperunit;  
 }  
 line = br.readLine();  
 }  
 *//Displaying the variables for Quantity.txt* pw1.println("Semi D is " + SemiD\_Quantity +" houses");  
 pw1.println("Bungalow is " + Bungalow\_Quantity +" houses");  
 pw1.println("Town House is " + TownHouse\_Quantity +" houses");  
 pw1.println("Double Storey is " + DoubleStorey\_Quantity +" houses");  
 pw1.println("Single Storey is " + SingleStorey\_Quantity +" houses");  
 *//Displaying the variables for total.txt* pw2.println("Mudra Sdn. Bhd : " + df.format(mudra));  
 pw2.println("salam Property : " + df.format(salam));  
 pw2.println("Creative Partner : " + df.format(creative));  
 pw2.println("Azlan Adik Beradik : " + df.format(azlan));  
 pw2.println("Nick & Rakan : " + df.format(nick));  
  
 pw1.close();  
 pw2.close();  
 br.close();  
 *//Error handling for possible file errors* } *catch* (FileNotFoundException fnf) {  
 System.***out***.print(fnf.getMessage());  
 } *catch* (EOFException ex) {  
 System.***out***.println(ex.getMessage());  
 } *catch* (IOException io) {  
 System.***out***.print(io.getMessage());  
 } *catch* (Exception nf) {  
 System.***out***.print(nf);  
 } *finally* {  
 System.***out***.println("System ends here... Bye Bye");  
 }  
 }  
}

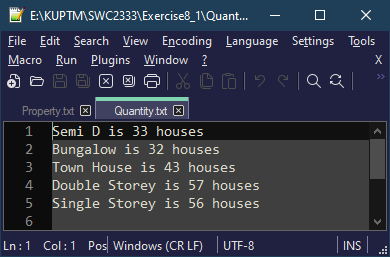
**Input :**

Property.txt

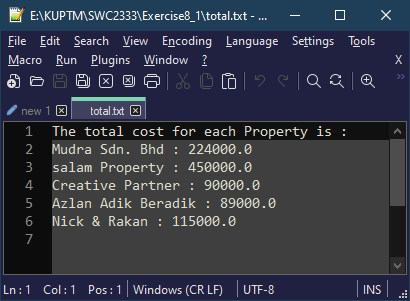


**Output :**

Quantity.txt



total.txt



**Lab Test Rubric**

**Student Name: Muhammad Alif Serbaini**  **Section No:01**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criterion** | **% of Grade** | **Excellent (100%)** | **Adequate (80%)** | **Poor (60%)** | **Not Met (0%)** |
| **Program Specifications / Correctness** | 50 | Program works correctly and meets all specification(s).  (41 – 50) | Some program functions work incorrect. Minor details of the program specification are violated.  (31-40) | Significant details of the specification are violated, program often exhibits incorrect behavior.  (21 – 30) | Program only functions correctly in very limited cases or not at all.  (0 – 20) |
| **Marks given** |  |  |  |  |  |
| **Input File** | 10 | The input file is read at the right path as coded in the program and correct data is inserted  (8 – 10) | The input file can be read and but little correct of data inserted  (5 – 7) | Evidence of input file and it cannot be read  (3 – 4) | No evidence of input file and no data is inserted  (0 – 2) |
| **Marks given** |  |  |  |  |  |
| **Readability** | 10 | code is clean, understandable, and well-organized  .  (8 – 10) | Minor issues with consistent indentation, use of whitespace, variable naming, or general organization.  (5 – 7) | At least one major issue with indentation, whitespace, variable names, or organization.  (3 – 4) | Major problems with at three or four of the readability subcategories.  (0 – 2) |
| **Marks given** |  |  |  |  |  |
| **Code Efficiency** | 10 | No errors, code uses the best approach in every case.  (8 – 10) | No error. Acceptable approach of code use*.*  (5 – 7) | Code uses poorly-chosen approaches in at least one place.  (3 – 4) | Code uses poorly-chosen approaches in all places.   (0 – 2) |
| **Marks given** |  |  |  |  |  |
| **Internal Documentation** | 10 | code is well- meaningful commented  (8 – 10) | code is *overly* commented.  (5 – 7) | code is lacking meaningful comments.  (3 – 4) | *un*commented very *lack* comment  (0 – 2) |
| **Marks given** |  |  |  |  |  |
| **Output** | 10 | Full correct output was described  (8 – 10) | Little correct evidence was described  (5 – 7) | Incorrect output  (3 – 4) | No evidence and testing section  (0 – 2) |
| **Marks given** |  |  |  |  |  |
| **Total** |  | **/ 100%** | | | |
| **Comment** |  | | | | |