OBJECT ORIENTED PROGRAMMING
CCS20704

Received Date :

Submission Due Date: 10 October 2024

Lecturer : Dr. Jamal Abdullahi

Weightage : 10%

Semester : SEPTEMBER 2024

Instruction to students:

- This is a GROUP assignment.
- Complete this cover sheet and attach it to your assignment (first page).

Student declaration:
I declare that: • ***This assignment is my/our own work.*** • ***I/we understand what is meant by plagiarism.*** • ***My lecturer has the right to deduct my marks in case of:*** - ***Late submission*** - ***Any plagiarism found in my assignment.***
NAME: MUHAMMAD HADIF IRFAN BIN MUHAMAD FAIZUL **MATRICS NO: 2082024020031** **PROGRAMME: BACHELOR IN COMPUTER SCIENCE** 

Attributes:

1. address (String) – The location of the house.
2. numberOfRooms (int) – Number of rooms in the house.
3. squareFeet (int) – The size of the house in square feet.
4. price (double) – The price of the house.
5. hasPool (boolean) – Whether the house has a pool.
6. isOccupied (boolean) – Whether the house is currently occupied.

Methods:

1. **Constructor** – Initializes the house's attributes.
2. **Getters and Setters** – For accessing and modifying the attributes.
3. **calculatePropertyValue()** – Calculates the property value based on its size, number of rooms, and pool presence.
4. **toggleOccupancy()** – Toggles the house's occupancy status (whether it's occupied or not).

Here's a Java implementation of the **House** class:

C: > Users > hadif > Desktop > Java OOP > House.java > Language Support for Java(TM) by Red Hat > House > displayDetails()

```java
class House { // House class representing a house with various attributes and methods
    // Private attributes (encapsulation)
    private String address;
    private int numberOfRooms;
    private int squareFeet;
    private double priceRM;
    private boolean hasPool;
    private boolean isOccupied;

    // Constructor to initialize the attributes
    public House(String address, int numberOfRooms, int squareFeet, double priceRM, boolean hasPool, boolean isOccupied) {
        this.address = address;
        this.numberOfRooms = numberOfRooms;
        this.squareFeet = squareFeet;
        this.priceRM = priceRM;
        this.hasPool = hasPool;
        this.isOccupied = isOccupied;
    }

    // Getter and setter methods for encapsulated attributes
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public int getNumberOfRooms() {
        return numberOfRooms;
    }

    public void setNumberOfRooms(int numberOfRooms) {
        this.numberOfRooms = numberOfRooms;
    }

    public int getSquareFeet() {
        return squareFeet;
    }

    public void setSquareFeet(int squareFeet) {
        this.squareFeet = squareFeet;
    }
```

J House.java 1 × J Main.java 1

C: > Users > hadif > Desktop > Java OOP > Assignment 1 house > J House.java > Language Support for Java(TM) by Red Hat > House

```
1   class House { // House class representing a house with various attributes and methods
44      public double getPriceRM() {
46      }
47
48      public void setPriceRM(double priceRM) {
49          this.priceRM = priceRM;
50      }
51
52      public boolean isHasPool() {
53          return hasPool;
54      }
55
56      public void setHasPool(boolean hasPool) {
57          this.hasPool = hasPool;
58      }
59
60      public boolean isOccupied() {
61          return isOccupied;
62      }
63
64      public void setOccupied(boolean isOccupied) {
65          this.isOccupied = isOccupied;
66      }
67
68      // Additional method to calculate property value based on attributes
69      public double calculatePropertyValue() {
70          double baseValue = squareFeet * 450; // Value per square foot in RM
71          if (hasPool) {
72              baseValue += 45000; // Increase value by RM 45,000 if there's a pool
73          }
74          return baseValue;
75      }
76
77      // Method to toggle the occupancy status
78      public void toggleOccupancy() {
79          if (isOccupied) {
80              System.out.println(x:"House is now vacant.");
81              isOccupied = false;
82          } else {
83              System.out.println(x:"House is now occupied.");
84              isOccupied = true; S
85          }
86      }
87
88      // Method to display house details
89      public void displayDetails() {
90          System.out.println("Address: " + address);
91          System.out.println("Number of rooms: " + numberOfRooms);
92          System.out.println("Square feet: " + squareFeet);
93          System.out.println("Price: RM " + priceRM);
94          System.out.println("Has a pool? " + (hasPool ? "Yes" : "No"));
95          System.out.println("Is occupied? " + (isOccupied ? "Yes" : "No"));
96          System.out.println("Estimated property value: RM " + calculatePropertyValue());
97          System.out.println();
98      }
```

Main class:

Main.java 1 | House.java 1

C: > Users > hadif > Desktop > Java OOP > Main.java > Java > Main

```java
public class Main { // Main class to interact with the House class
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create a new House object with price in RM
        House myHouse = new House(address:"123 Maple Street", numberOfRooms:4, squareFeet:2500, priceRM:1200000, ha…true, false);

        // Print the house details
        System.out.println(x:"House Details:");
        myHouse.displayDetails();

        // Toggle occupancy status
        System.out.println(x:"Toggling occupancy...");
        myHouse.toggleOccupancy();

        // Print updated details
        System.out.println(x:"\nUpdated House Details:");
        myHouse.displayDetails();
    }
}
```

Output:

PROBLEMS 2 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

powershell | Run: Main

```
PS C:\Users\hadif>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.4.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\hadif\AppData\Local\Temp\vscodesws_43dfb\jdt_ws\jdt.ls-java-project\bin' 'Main'
House Details:
Address: 123 Maple Street
Number of rooms: 4
Square feet: 2500
Price: RM 1200000.0
Has a pool? Yes
Is occupied? No
Estimated property value: RM 1170000.0

Toggling occupancy...
House is now occupied.

Updated House Details:
Address: 123 Maple Street
Number of rooms: 4
Square feet: 2500
Price: RM 1200000.0
Has a pool? Yes
Is occupied? Yes
Estimated property value: RM 1170000.0
```

Java: Ready

NAME: MUHAMMAD HADIF IRFAN BIN MUHAMAD FAIZUL
MATRICS NO: 2082024020031

RUBRIC FOR ASSIGNMNET 1

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)	Marks
Object Design (20%)	Object has at least 6 attributes, well-named, and appropriately typed. Object has at least 4 well-defined and well-documented methods. Proper use of constructor, setters, and getters with clear functionality and documentation.	Object has at least 5 attributes, mostly well-named and appropriately typed. Object has at least 4 well-defined and documented methods. Adequate use of constructor, setters, and getters.	Object has less than 5 attributes, some not well-named or typed. Object has less than 4 methods or lacking documentation. Limited or unclear use of constructor, setters, and getters.	Object design is incomplete or missing significant attributes or methods. Poor or missing use of constructor, setters, and getters.	20
Encapsulation (20%)	Excellent use of encapsulation with clear visibility modifiers and access control. All data is properly encapsulated with no direct access to attributes. Proper use of accessors and mutators.	Good use of encapsulation with mostly clear visibility modifiers and access control. Most data is encapsulated, with minimal direct access to attributes. Adequate use of accessors and mutators.	Some issues with encapsulation and access control, but it generally prevents unauthorized access. Limited use of accessors and mutators.	Poor or no use of encapsulation, allowing direct access to attributes. Lack of accessors and mutators.	20
Main Class Implementation (20%)	Effective and creative use of control structures to manipulate the object's attributes and methods Code is well-organized, easy to read, and free from errors. Exceptional demonstration of programming skills.	Good use of control structures to manipulate the object, with some creativity. Code is mostly organized, readable, and has minor errors. Demonstrates solid programming skills.	Adequate use of control structures, but the manipulation lacks creativity or completeness. Code organization and readability need improvement. Basic demonstration of programming skills.	Incomplete or minimal use of control structures, leading to limited object manipulation. Poor code organization, readability, and numerous errors. Lack of programming skills demonstrated.	15

NAME: MUHAMMAD HADIF IRFAN BIN MUHAMAD FAIZUL
MATRICS NO: 2082024020031

Creativity and Object Choice (20%)	Highly creative and unique choice of object, showing deep understanding of OOP principles. Innovative use of attributes and methods that go beyond the basic requirements. Demonstrates exceptional creativity.	Creative choice of object, demonstrating an understanding of OOP principles. Some innovative use of attributes and methods. Shows creativity.	Object choice is somewhat conventional, with limited creativity. Attributes and methods are basic and meet minimum requirements. Limited creativity.	Lack of creativity in object choice or generic selection. Attributes and methods are basic and lack creativity. No demonstration of creative thinking.	20
Documentation and Comments (10%)	Code is extensively documented with clear explanations for each attribute, method, and control structure. Comments are used effectively to provide insights into the code's functionality. Excellent documentation.	Code is adequately documented with explanations for most attributes, methods, and control structures. Some comments are used to clarify code functionality. Good documentation.	Limited documentation with explanations for only a few attributes, methods, or control structures. Minimal use of comments. Basic documentation.	Code is poorly or not documented, with little or no explanation of attributes, methods, or control structures. Lack of comments. Inadequate documentation.	10
REMARKS Add getter and setter in the main method					10 95
TOTAL MARKS					