OBJECT ORIENTED PROGRAMMING
CCS20704

Received Date	**:**
Submission Due Date:	**10 October 2024**
Lecturer	**: Dr. Jamal Abdullahi**
Weightage	**: 10%**
Semester	**: SEPTEMBER 2024**

Instruction to students:

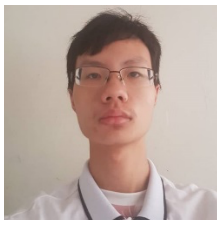
- This is a GROUP assignment.
- Complete this cover sheet and attach it to your assignment (first page).

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: Basil Kwa Shao Rong
MATRICS NO: 012022090888
PROGRAMME: BICT



ASSIGNMENT 1

Individual

INSTRUCTION:

1. Students are required to submit a report **(LIGHT BLUE**).
2. This assignment contributes 10% of the overall assessment.
3. Upload a report and a zipped project directory with all codes and resources required to compile and run the code in eklas.
4. Print your report (codes & sample of output) together with cover page and all the rubrics.
   Last date of submission: 10 October 2024.

COURSE LEARNING OUTCOME:

CO2 Apply the concepts of Object-oriented Programming in solving computing problems using Java. (C3, PLO2)

QUESTION

Task:
You are required to **design a class** for a chosen object, ensuring that it adheres to the **principles of object-oriented programming (OOP)**, especially **encapsulation**. This class should include **six attributes** and **four methods**, consisting of:

- **One constructor**
- **Getter and Setter methods** for at least some attributes
- Additional methods to manipulate or interact with the object's attributes

Class Requirements:

1. **Attributes:**
   - ◊ A minimum of **six attributes**, each representing a unique property of the object (for example, color, size, weight, etc.).
   - ◊ Attributes must be encapsulated using **private** access modifiers, and you should provide **getter and setter methods** to access and modify the attributes safely.
2. **Methods:**
   - ◊ At least **four methods**, including:
     - **A constructor** to initialize the object's attributes.
     - **Getter and Setter methods** to access and modify the encapsulated attributes.
     - At least one **other method** that provides functionality based on the object's attributes (e.g., a method to calculate something based on the object's properties).

Main Class:

- Create a **main class** to interact with your chosen object's attributes and methods.
- Your main class should demonstrate the use of:
  - ◊ **Control structures**, such as:
    - **Selection (if statements)** to make decisions based on object attributes
    - **Looping (for or while loops)** to iterate or repeatedly interact with the object

- ◊ **Manipulation of attributes and methods** using getter and setter methods.

Additional Notes:

- ◊ Your code should be **well-documented** and **commented**, explaining each attribute and method's purpose and functionality.
- ◊ The evaluation of your assignment will be based on:
  - Correctness and **functionality** of your code
  - Creativity in the **choice of object**
  - The effective use of **control structures** to manipulate the object's data
  - Clarity and **quality of your code documentation**

Submission Notes:

Ensure you submit the following:

1. **Class files** for each group member's object.
2. **The main class** file that demonstrates interaction with the objects.
3. **Code comments** that clearly explain the purpose of each class, method, and control structure used.

Object > Car

Main class

```
public class Main {
   public static void main(String[] args) {
      Car myCar = new Car("Tesla", "Model S", 2022, "Red", 5000.0, true);

      myCar.displayCarDetails();

      if (myCar.isAntique()) {
         System.out.println("The car is an antique.");
      } else {
         System.out.println("The car is not an antique.");
      }

      myCar.setMileage(6000.0);
      myCar.setColor("Blue");

      myCar.displayCarDetails();
   }
}



public class Car {
   private String brand;
   private String model;
   private int year;
   private String color;
   private double mileage;
   private boolean isElectric;

   public Car(String brand, String model, int year, String color, double mileage, boolean
isElectric) {
      this.brand = brand;
      this.model = model;
      this.year = year;
      this.color = color;
      this.mileage = mileage;
      this.isElectric = isElectric;
   }

   public String getBrand() {
      return brand;
   }

   public void setBrand(String brand) {
      this.brand = brand;
   }

   public String getModel() {
      return model;
   }

   public void setModel(String model) {
      this.model = model;
   }

   public int getYear() {
      return year;
   }

   public void setYear(int year) {
      this.year = year;
   }

   public String getColor() {
      return color;
   }

   public void setColor(String color) {
      this.color = color;
   }

   public double getMileage() {
      return mileage;
   }

   public void setMileage(double mileage) {
      this.mileage = mileage;
   }

   public boolean isElectric() {
      return isElectric;
   }

   public void setElectric(boolean isElectric) {
      this.isElectric = isElectric;
   }

   public void displayCarDetails() {
      System.out.println("Car Details: ");
      System.out.println("Brand: " + brand);
      System.out.println("Model: " + model);
```

```
        System.out.println("Year: " + year);
        System.out.println("Color: " + color);
        System.out.println("Mileage: " + mileage + " km");
        System.out.println("Is Electric: " + (isElectric ? "Yes" : "No"));
    }

    public boolean isAntique() {
        int currentYear = java.time.Year.now().getValue();
        return (currentYear - year) > 25;
    }
}
```

RUBRIC FOR ASSIGNMNET 1

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)	Marks
Object Design (20%)	Object has at least 5 attributes, well-named, and appropriately typed. Object has at least 5 well-defined and well-documented methods. Proper use of constructor, setters, and getters with clear functionality and documentation.	Object has at least 4 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 4 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.	15
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.	15
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.	5

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.	5
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.	0
REMARKS					10 50
TOTAL MARKS					

END OF QUESTION