OBJECT ORIENTED PROGRAMMING
CCS20704

Received Date	**: 02 October 2024**
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: NUR FATWA HIDAYAH BINTI MOHD NOR AZIZAN
MATRICS NO: 2082024090008
PROGRAMME: BACHELOR IN COMPUTER SCIENCE (BCS)

In this case study, I have chosen to design a class representing a Smartphone. The smartphone is a widely used object in modern society and its functionality and attributes make it an ideal candidate for demonstrating object-oriented programming (OOP) principles, particularly encapsulation. By creating a class to represent a smartphone, I aim to capture essential characteristics and behaviors, including brand, model, storage capacity, RAM, battery life and screen size. These attributes are encapsulated, meaning they are only accessible through carefully designed methods to ensure the integrity and security of the data. The class will include six key attributes that represent essential properties of the smartphone:

- **Brand:** This attribute captures the manufacturer of the smartphone, such as Apple or Samsung, providing a way to identify the product's maker.
- **Model:** The specific name or number of the phone model giving a precise reference to the product variant.
- **Storage Capacity:** Measured in gigabytes (GB), this attribute represents the internal storage space available for user data, apps, and media, directly influencing the phone's ability to store large files and applications.
- **RAM:** The amount of random-access memory (RAM) in gigabytes which is crucial for determining the phone's multitasking ability and overall performance.
- **Battery Capacity:** This defines the total capacity of the phone’s battery in milliamp hours (mAh), a critical attribute that affects how long the phone can operate before needing a recharge, especially under heavy usage conditions.
- **Screen Size:** Measured in inches, the screen size represents the diagonal length of the phone’s display, impacting the device's usability, media experience and overall form factor.

The class will feature a constructor to initialize the smartphone object with specific values for the defined attributes, ensuring that the object is properly set up when created. Getter and setter methods will be used to access and modify the attributes safely, allowing controlled access to sensitive data like the brand, model, storage capacity and battery capacity. An additional method, calculateBatteryLife(), will compute the estimated battery life based on the usage rate of the phone in milliamp hours (mAh) per hour. This method helps users understand how long their phone battery can last under certain usage conditions. Another method, displaySmartphoneDetails(), will print out all the attributes of the smartphone in a clear and user-friendly format which allows the user to view the phone's specifications and performance details at a glance. This method simplifies the interaction by giving a concise overview of the smartphone's features.

Each of these attributes not only defines the technical specifications of the smartphone but also plays a key role in its functionality and user experience, making them integral to the class design. The main class will demonstrate how these attributes and methods interact. The program will allow the user to view and update the smartphone's attributes such as changing the brand or storage capacity. It will also calculate and display how long the smartphone's battery will last be based on the user's input regarding battery usage. Additionally, a decision-making structure using an if statement will inform the user whether the battery life is sufficient for typical daily use. In summary, the program will display the smartphone's details, update certain attributes, and calculate battery life, ensuring both technical correctness and user engagement.

THE CODE

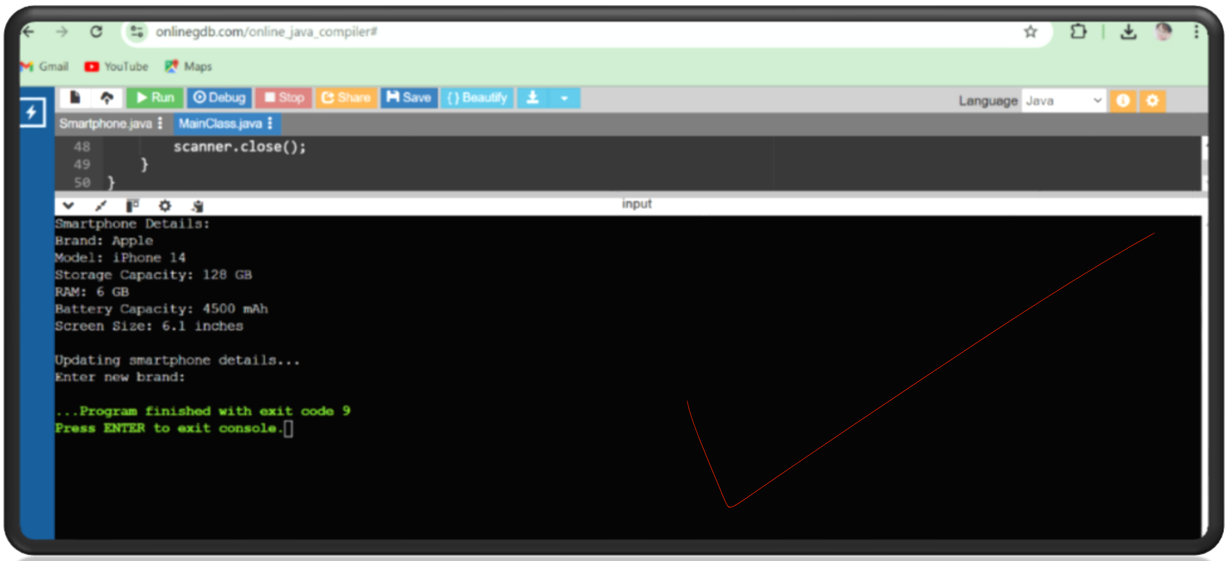
```
// Class representing a Smartphone object
public class Smartphone {
    // Private attributes (encapsulated)
    private String brand;
    private String model;
    private int storageCapacity; // in GB
    private int ram; // in GB
    private int batteryCapacity; // in mAh
    private double screenSize; // in inches

    // Constructor to initialize the object
    public Smartphone(String brand, String model, int
storageCapacity, int ram, int batteryCapacity, double
screenSize) {
        this.brand = brand;
        this.model = model;
        this.storageCapacity = storageCapacity;
        this.ram = ram;
        this.batteryCapacity = batteryCapacity;
        this.screenSize = screenSize;
    }

    // Getter and Setter for brand
    public String getBrand() {
        return brand;
    }
    Public void setBrand(String brand) {
        this.brand = brand;
    }
```

```java
// Getter and Setter for model
public String getModel() {
    return model;
}
public void setModel(String model) {
    this.model = model;
}

// Getter and Setter for storageCapacity
public int getStorageCapacity() {
    return storageCapacity;
}
public void setStorageCapacity(int storageCapacity) {
    this.storageCapacity = storageCapacity;
}

// Getter and Setter for ram
public int getRam() {
    return ram;
}
public void setRam(int ram) {
    this.ram = ram;
}

// Getter and Setter for batteryCapacity
public int getBatteryCapacity() {
    return batteryCapacity;
}
public void setBatteryCapacity(int batteryCapacity) {
    this.batteryCapacity = batteryCapacity;
}
```
✓
✓
✓
✓

```
    // Getter and Setter for screenSize
    public double getScreenSize() {
        return screenSize;
    }
    public void setScreenSize(double screenSize) {
        this.screenSize = screenSize;
    }

    // Method to calculate approximate battery life based on
usage
    public double calculateBatteryLife(int usagePerHour) {
        return (double) batteryCapacity / usagePerHour; //
hours of usage based on usage rate
    }

    // Method to display the details of the smartphone
    public void displaySmartphoneDetails() {
        System.out.println("Smartphone Details:");
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
        System.out.println("Storage Capacity: " +
storageCapacity + " GB");
        System.out.println("RAM: " + ram + " GB");
        System.out.println("Battery Capacity: " +
batteryCapacity + " mAh");
        System.out.println("Screen Size: " + screenSize + "
inches");
    }
}
```

```
// Main class to interact with Smartphone objects
import java.util.Scanner;
public class MainClass {
    public static void main(String[] args) {

        // Create a Scanner object to read input
        Scanner scanner = new Scanner(System.in);

        // Create a Smartphone object using the constructor
        Smartphone myPhone = new Smartphone("Apple", "iPhone
14", 128, 6, 4500, 6.1);

        // Display initial smartphone details
        myPhone.displaySmartphoneDetails();

        // Interact with attributes using getter and setter
methods
        System.out.println("\nUpdating smartphone
details...");

        // Update the brand
        System.out.print("Enter new brand: ");
        String newBrand = scanner.nextLine();
        myPhone.setBrand(newBrand);

        // Update the storage capacity
        System.out.print("Enter new storage capacity (GB): ");
        int newStorage = scanner.nextInt();
        myPhone.setStorageCapacity(newStorage);
        // Display updated smartphone details
        System.out.println("\nUpdated smartphone details:");
        myPhone.displaySmartphoneDetails();

        // Control structure: Check if battery is sufficient
for usage
        System.out.print("\nEnter usage rate per hour (mAh):
");
        int usageRate = scanner.nextInt();

        // Calculate and display estimated battery life
        double estimatedBatteryLife =
myPhone.calculateBatteryLife(usageRate);
        System.out.println("Estimated battery life: " +
estimatedBatteryLife + " hours");

        // Decision-making using if-else statement
        if (estimatedBatteryLife > 5) {
            System.out.println("Battery life is sufficient for
a day's use.");
        } else {
            System.out.println("You might need to recharge
your phone soon.");
        }

        // Close the scanner
        scanner.close();
    }
}
```

THE OUTPUT



```
        scanner.close();
    }
}
```

```
Smartphone Details:
Brand: Apple
Model: iPhone 14
Storage Capacity: 128 GB
RAM: 6 GB
Battery Capacity: 4500 mAh
Screen Size: 6.1 inches

Updating smartphone details...
Enter new brand:

...Program finished with exit code 9
Press ENTER to exit console.
```

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

msu management & science university

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.	17
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 Simplify the code and add getter and setter in the main method also in the display method use the getter methods					
TOTAL MARKS					93

10

END OF QUESTION