WOLAITA SODO UNIVERSITY

OBJECT-ORIENTED-PROGRAMMING

PROJECT

COURSE TITLE – OBJECT ORIENTED PROGRAMMING

COURCE TITLE – COSC2051

DEPARTMENT- COMPUTER SCIENCE

GROUP MEMBERS

NO	NAME	ID NUMBER
1	DAGIM DAWIT	UGR/92584/16
2	ASHENAFI DEGIF	UGR/91463/16
3	DEGNESH DALGA	UGR/91619/16
4	MAHILET AMSALU	UGR/91944/16
5	WUBDEL ASEFA	UGR/98813/16
6	TIZIBT SHANBEL	UGR/92350/16

SUBMISSSION DATE – 20/05/2017 E.C
INSTRUCTOR -- DAWIT UTA (M.TECH.)

Assignment 1: Student Grading System

Problem:

Create a class Student with the following attributes and methods:

- **Attributes:** name, rollNumber, marks (list of marks in 5 subjects)
- Methods:
    1. calculateTotal(): Return the total marks.
    2. calculateAverage(): Return the average marks.
    3. displayResult(): Display the total, average, and grade based on the average (A for average >= 90, B for average >= 75, etc.).

Instructions:

1. Create at least three Student objects.
2. Display the result for each student.

Answer:

```
class Student {
    // Attributes
    String name;
    int rollNumber;
    int[] marks;

    // Constructor to initialize the student details
    public Student(String name, int rollNumber, int[] marks) {
        this.name = name;
        this.rollNumber = rollNumber;
        this.marks = marks;
        // Debugging: Print to check if constructor is working
        System.out.println("Student: " + name);
        System.out.println("Roll Number: " + rollNumber);
        System.out.print("Marks: ");
        for (int mark : marks) {
            System.out.print(mark + " ");
        }
        System.out.println( );
        System.out.println( );
```

```
}

    // Method to calculate total marks
    public int calculateTotal() {
        int total = 0;
        for (int mark : marks) {
            total += mark;
        }
        return total;
    }

    // Method to calculate average marks
    public double calculateAverage() {
        return calculateTotal() / (double) marks.length;
    }

    // Method to display result
    public void displayResult() {
        int total = calculateTotal();
        double average = calculateAverage();
        char grade;

        // Determine the grade based on the average
        if (average >= 90) {
            grade = 'A';
        } else if (average >= 75) {
            grade = 'B';
        } else if (average >= 60) {
            grade = 'C';
        } else if (average >= 45) {
            grade = 'D';
        } else {
            grade = 'F';
        }
```

```
        // Display the result
        System.out.println("Student: " + name);
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Total Marks: " + total);
        System.out.println("Average Marks: " + String.format("%.2f", average));
        System.out.println("Grade: " + grade);
        System.out.println("-----------------------------");
    }}
public class StudentTest {
    public static void main(String[] args) {
                // Create at least three Student objects with marks initialized correctly
                int[] marks1 = {85, 90, 88, 92, 87};
                Student student1 = new Student("Dagim", 101, marks1);

                int[] marks2 = {75, 78, 80, 70, 69};
                Student student2 = new Student("Ashenafi", 102, marks2);

                int[] marks3 = {60, 55, 62, 58, 65};
                Student student3 = new Student("Degnesh", 103, marks3);
                // Display the result for each student
                student1.displayResult();
                student2.displayResult();
                student3.displayResult();
    }}
```

output:

Student: Dagim
Roll Number: 101
Marks: 85 90 88 92 87

Student: Ashenafi
Roll Number: 102
Marks: 75 78 80 70 69

Student: Degnesh
Roll Number: 103
Marks: 60 55 62 58 65

Student: Dagim
Roll Number: 101
Total Marks: 442
Average Marks: 88.40
Grade: B

Student: Ashenafi
Roll Number: 102
Total Marks: 372
Average Marks: 74.40
Grade: C

Student: Degnesh
Roll Number: 103
Total Marks: 300
Average Marks: 60.00
Grade: C

Assignment 2: Employee Management System

Problem:

Create a base class Employee with attributes name, employeeID, and salary.
Derive two classes from Employee:
1. Manager with an additional attribute bonus.
2. Developer with an additional attribute technology.

Instructions:

1. Implement methods in each class to display the employee's details.
2. Create objects for both Manager and Developer, and call the methods to display their details.

Answer:

```
/*Here's how you can implement this problem in Java, using inheritance where Manager and Developer classes derive from the base Employee class:*/

// Base class Employee
class Employee {
    // Attributes for Employee class
    String name;
    int employeeID;
    double salary;

    // Constructor to initialize Employee details
    public Employee(String name, int employeeID, double salary) {
        this.name = name;
        this.employeeID = employeeID;
        this.salary = salary;
    }
    // Method to display employee details
    public void displayDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + employeeID);
        System.out.println("Salary: $" + salary);
    }
 }

// Derived class Manager
class Manager extends Employee {
    // Additional attribute for Manager class
    double bonus;

    // Constructor to initialize Manager details (includes Employee details)
    public Manager(String name, int employeeID, double salary, double bonus) {
        super(name, employeeID, salary); // Calling the Employee class constructor
```

```
        this.bonus = bonus;
    }

    // Method to display Manager's details, extending the base class method
    @Override
    public void displayDetails() {
        super.displayDetails(); // Call Employee's displayDetails()
        System.out.println("Bonus: $" + bonus);
        System.out.println("-----------------------------");
    }
}

// Derived class Developer
class Developer extends Employee {
    // Additional attribute for Developer class
    String technology;
    // Constructor to initialize Developer details (includes Employee details)
    public Developer(String name, int employeeID, double salary, String technology) {
        super(name, employeeID, salary); // Calling the Employee class constructor
        this.technology = technology;
    }

    // Method to display Developer's details, extending the base class method
    @Override
    public void displayDetails() {
        super.displayDetails(); // Call Employee's displayDetails()
        System.out.println("Technology: " + technology);
        System.out.println("-----------------------------");
    }
}

public class Main {
    public static void main(String[] args) {
        // Create Manager object and display details
        Manager manager = new Manager("Mahilet", 101, 95000, 5000);
```

```
        manager.displayDetails();

        // Create Developer object and display details
        Developer developer = new Developer("Wubdel", 102, 80000, "Java");
        developer.displayDetails();
    }
}
```

output:

```
Employee Name: Mahilet
Employee ID: 101
Salary: $95000.0
Bonus: $5000.0
----------------------------
Employee Name: Wubdel
Employee ID: 102
Salary: $80000.0
Technology: Java
----------------------------
```

Assignment 3: Library System

Problem:

Create a class Library **with the following methods:**

- addBook(title, author): **Add a book to the library.**
- borrowBook(title): **Mark a book as borrowed if it is available.**
- returnBook(title): **Mark a book as returned.**
- displayBooks(): **Display the list of available books.**

Instructions:

1. Create a Library object.
2. Add books, borrow a book, return a book, and display the list of books at each step.

Answer:

```
import java.util.ArrayList;
```

```
import java.util.List;

        class Library {
    // List to store books in the library
    private List<Book> books;

    // Constructor to initialize the library with an empty list of books
    public Library() {
        books = new ArrayList<>();
    }

    // Method to add a book to the library
    public void addBook(String title, String author) {
        Book book = new Book(title, author, true); // Book is available when added
      books.add(book);
        System.out.println("Book added: " + title + " by " + author);
    }

    // Method to borrow a book from the library
    public void borrowBook(String title) {
        for (Book book : books) {
            if (book.getTitle().equals(title) && book.isAvailable()) {
                book.setAvailable(false); // Mark the book as borrowed
                System.out.println("You have borrowed the book: " + title);
                return;
            }
        }
        System.out.println("The book \"" + title + "\" is not available.");
    }

    // Method to return a book to the library
    public void returnBook(String title) {
        for (Book book : books) {
            if (book.getTitle().equals(title) && !book.isAvailable()) {
                book.setAvailable(true); // Mark the book as returned
```

```
            System.out.println("You have returned the book: " + title);
            return;
        }
    }
    System.out.println("The book \"" + title + "\" was not borrowed from this library.");
}

// Method to display the list of available books in the library
public void displayBooks() {
    System.out.println("\nAvailable books in the library:");
    for (Book book : books) {
        if (book.isAvailable()) {
            System.out.println(book.getTitle() + " by " + book.getAuthor());
        }
    }
    System.out.println("------------------------------");
}

// Inner class to represent a Book
class Book {
    private String title;
    private String author;
    private boolean available;

    public Book(String title, String author, boolean available) {
        this.title = title;
        this.author = author;
        this.available = available;
    }
    public String getTitle() {
        return title;
    }
    public String getAuthor() {
        return author;
    }
```

```
    public boolean isAvailable() {
        return available;
    }
    public void setAvailable(boolean available) {
        this.available = available;
    }
}}


public class Main {

    public static void main(String[] args) {
    // Create a Library object

    Library library = new Library();

    // Add books to the library

    library.addBook("The Great Gatsby", "F. Scott Fitzgerald");
    library.addBook("1984", "George Orwell");
    library.addBook("To Kill a Mockingbird", "Harper Lee");

    // Display available books
    library.displayBooks();
    // Borrow a book
    library.borrowBook("1984");
    // Display available books after borrowing
    library.displayBooks();
    // Return a book
    library.returnBook("1984");
    // Display available books after returning
    library.displayBooks();
    }
}
```

output:

Book added: The Great Gatsby by F. Scott Fitzgerald
Book added: 1984 by George Orwell
Book added: To Kill a Mockingbird by Harper Lee

Available books in the library:
The Great Gatsby by F. Scott Fitzgerald
1984 by George Orwell
To Kill a Mockingbird by Harper Lee

You have borrowed the book: 1984

Available books in the library:
The Great Gatsby by F. Scott Fitzgerald
To Kill a Mockingbird by Harper Lee

You have returned the book: 1984

Available books in the library:
The Great Gatsby by F. Scott Fitzgerald
1984 by George Orwell
To Kill a Mockingbird by Harper Lee

Assignment 4: Vehicle Rental System

Problem:

Design a vehicle rental program using the following classes:

1. Vehicle (base class) with attributes vehicleID, brand, pricePerDay
2. Car and Bike (derived classes) with additional attributes like numberOfSeats and engineCapacity respectively.

Implement a method rentVehicle(days) in each class to calculate the total rental cost.

Answer:

```
class Vehicle {
    // Attributes common to all vehicles
    int vehicleID;
    String brand;
    double pricePerDay;
    // Constructor to initialize the common attributes
    public Vehicle(int vehicleID, String brand, double pricePerDay) {
        this.vehicleID = vehicleID;
        this.brand = brand;
        this.pricePerDay = pricePerDay;
    }
    // Method to calculate the rental cost (for base class, can be overridden by derived classes)
    public double rentVehicle(int days) {
        return pricePerDay * days;
    }
    // Method to display vehicle details
    public void displayDetails() {
        System.out.println("Vehicle ID: " + vehicleID);
        System.out.println("Brand: " + brand);
        System.out.println("Price per Day: $" + pricePerDay);
    }
}

class Car extends Vehicle {
    // Additional attribute for Car class
    int numberOfSeats;
    // Constructor to initialize Car details (includes Vehicle details)
    public Car(int vehicleID, String brand, double pricePerDay, int numberOfSeats) {
        super(vehicleID, brand, pricePerDay); // Calling the Vehicle class constructor
        this.numberOfSeats = numberOfSeats;
    }

    // Overriding the rentVehicle method to include Car-specific details (if needed)
    @Override
    public double rentVehicle(int days) {
```

```
        double totalCost = super.rentVehicle(days); // Base cost from Vehicle class
        // Optionally, you can add a surcharge for cars with more seats
        if (numberOfSeats > 5) {
            totalCost += 10 * days; // Additional cost for larger cars
        }
        return totalCost;
    }

    // Method to display car details (in addition to Vehicle details)
    @Override
    public void displayDetails() {
        super.displayDetails(); // Display Vehicle details
        System.out.println("Number of Seats: " + numberOfSeats);
    }
}

//Derived class Bike
class Bike extends Vehicle {
 // Additional attribute for Bike class
 double engineCapacity; // Engine capacity in CC
 // Constructor to initialize Bike details (includes Vehicle details)
 public Bike(int vehicleID, String brand, double pricePerDay, double engineCapacity) {
     super(vehicleID, brand, pricePerDay); // Calling the Vehicle class constructor
     this.engineCapacity = engineCapacity;
 }

 // Overriding the rentVehicle method to include Bike-specific details
 @Override
 public double rentVehicle(int days) {
     double totalCost = super.rentVehicle(days); // Base cost from Vehicle class
     // Optionally, you can add a surcharge for larger engine capacity
     if (engineCapacity > 500) {
         totalCost += 5 * days; // Additional cost for bikes with larger engine capacity
     }
     return totalCost;
```

```
}

// Method to display bike details (in addition to Vehicle details)
@Override
public void displayDetails() {
    super.displayDetails(); // Display Vehicle details
    System.out.println("Engine Capacity: " + engineCapacity + " CC");
}
}

public class Main {
        public static void main(String[] args) {

                // Create a Car object
        Car car = new Car(1, "Toyota", 50.0, 7);

        // Create a Bike object
        Bike bike = new Bike(2, "Harley Davidson", 30.0, 750);

        // Display details and rent the vehicle for 5 days

        System.out.println("Car Details:");
        car.displayDetails();
        System.out.println("Total Rental Cost for 5 days: $" + car.rentVehicle(5));
        System.out.println("-----------------------------");
        System.out.println("Bike Details:");
        bike.displayDetails();
        System.out.println("Total Rental Cost for 5 days: $" + bike.rentVehicle(5));
          }}
```

output:

Car Details:

Vehicle ID: 1

Brand: Toyota

Price per Day: $50.0

Number of Seats: 7

Total Rental Cost for 5 days: $300.0

Bike Details:

Vehicle ID: 2

Brand: Harley Davidson

Price per Day: $30.0

Engine Capacity: 750.0 CC

Total Rental Cost for 5 days: $175.0