

Moiz Ahmed

Oop

Bel 4

Task 1

```
class Employee {
    private String name;
    private String address;
    private double salary;

    public Employee(String name, String address, double salary) {
        this.name = name;
        this.address = address;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public double getSalary() {
        return salary;
    }

    public double calculateBonus() {
        return 0; // Base class bonus calculation, overridden in subclasses
    }
}

class Manager extends Employee {
    private double bonusPercentage = 0.05; // 5% bonus for managers

    public Manager(String name, String address, double salary) {
        super(name, address, salary);
    }

    @Override
    public double calculateBonus() {
        return getSalary() * bonusPercentage;
    }
}
```

```

40     ,
41
42     class Developer extends Employee {
43         private double bonusPercentage = 0.10; // 10% bonus for developers
44
45         public Developer(String name, String address, double salary) {
46             super(name, address, salary);
47         }
48
49         @Override
50         public double calculateBonus() {
51             return getSalary() * bonusPercentage;
52         }
53     }
54
55     class Programmer extends Employee {
56         private double bonusPercentage = 0.15; // 15% bonus for programmers
57
58         public Programmer(String name, String address, double salary) {
59             super(name, address, salary);
60         }
61
62         @Override
63         public double calculateBonus() {
64             return getSalary() * bonusPercentage;
65         }
66     }
67
68     public class CompanyEmployees {
69         public static void main(String[] args) {
70             Manager manager = new Manager("John Doe", "123 Main St", 50000);
71             Developer developer = new Developer("Jane Smith", "456 Oak St", 60000);
72             Programmer programmer = new Programmer("Alice Johnson", "789 Elm St", 70000);
73
74             System.out.println("Manager Bonus: $" + manager.calculateBonus());
75             System.out.println("Developer Bonus: $" + developer.calculateBonus());
76             System.out.println("Programmer Bonus: $" + programmer.calculateBonus());
77         }

```

## Task 2

```
1 // Base class File
2 class File {
3     protected String name;
4     protected int width;
5     protected int height;
6
7     public File(String name, int width, int height) {
8         this.name = name;
9         this.width = width;
10        this.height = height;
11    }
12
13    public void displayDetails() {
14        System.out.println("File Name: " + name);
15        System.out.println("Width: " + width);
16        System.out.println("Height: " + height);
17    }
18 }
19
20 // Subclass ImageFile
21 class ImageFile extends File {
22     private int bitsPerPixel;
23
24     public ImageFile(String name, int width, int height, int bitsPerPixel) {
25         super(name, width, height);
26         this.bitsPerPixel = bitsPerPixel;
27     }
28
29     @Override
30     public void displayDetails() {
31         super.displayDetails();
32         System.out.println("Bits Per Pixel: " + bitsPerPixel);
33     }
34 }
35
36 // Subclass VideoFile
```

```

36 // Subclass VideoFile
37 class VideoFile extends File {
38     private int framePerSecond;
39     private int duration;
40
41     public VideoFile(String name, int width, int height, int framePerSecond, int duration) {
42         super(name, width, height);
43         this.framePerSecond = framePerSecond;
44         this.duration = duration;
45     }
46
47     @Override
48     public void displayDetails() {
49         super.displayDetails();
50         System.out.println("Frame Per Second: " + framePerSecond);
51         System.out.println("Duration: " + duration + " seconds");
52     }
53 }
54
55 // Test class to generate files and display their details
56 public class FileManager {
57     public static void main(String[] args) {
58         // Creating an image file
59         ImageFile image = new ImageFile("example.jpg", 1920, 1080, 24);
60         System.out.println("Image File Details:");
61         image.displayDetails();
62         System.out.println();
63
64         // Creating a video file
65         VideoFile video = new VideoFile("example.mp4", 1920, 1080, 30, 120);
66         System.out.println("Video File Details:");
67         video.displayDetails();
68     }
59 }

```

#### Output - Run (mavenproject2)

```

cd C:\Users\Laptop inn\Documents\NetBeansProjects\mavenproject2; "JAVA_HOME
Running NetBeans Compile On Save execution. Phase execution is skipped and
Scanning for projects...

-----< com.mycompany:mavenproject2 >-----
Building mavenproject2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject2 ---
Hello World!

BUILD SUCCESS

Total time: 1.851 s
Finished at: 2024-04-26T15:18:50+05:00

```

### Task 3

```
1  class Employee {
2      private double salary;
3      private int numberOfHours;
4
5      // Constructor to initialize salary and number of hours
6      public Employee(double salary, int numberOfHours) {
7          this.salary = salary;
8          this.numberOfHours = numberOfHours;
9      }
10
11     // Method to add $10 to salary if it's less than $500
12     public void addSal() {
13         if (salary < 500) {
14             salary += 10;
15         }
16     }
17
18     // Method to add $5 to salary if the number of hours is more than 6
19     public void addWork() {
20         if (numberOfHours > 6) {
21             salary += 5;
22         }
23     }
24
25     // Method to calculate and return the final salary
26     public double calculateFinalSalary() {
27         return salary;
28     }
29
30     // Method to get employee information
31     public void getInfo(double salaryPerHour, int numberOfHours) {
32         this.salary = salaryPerHour * numberOfHours;
33         this.numberOfHours = numberOfHours;
34     }
35 }
36
37 public class Main {
38     public static void main(String[] args) {
```

```

38 public static void main(String[] args) {
39     // Create an Employee object
40     Employee employee = new Employee(0, 0);
41
42     // Get user input for salary per hour and number of hours
43     double salaryPerHour = 0;
44     int numberOfHours = 0;
45
46     // Sample user input (you can replace this with actual user input code)
47     salaryPerHour = 12.5; // Sample salary per hour
48     numberOfHours = 8; // Sample number of hours worked
49
50     // Call getInfo() method to set employee information
51     employee.getInfo(salaryPerHour, numberOfHours);
52
53     // Call addSal() method to add $10 to salary if it's less than $500
54     employee.addSal();
55
56     // Call addWork() method to add $5 to salary if number of hours is more than 6
57     employee.addWork();
58
59     // Print final salary
60     System.out.println("Final Salary: $" + employee.calculateFinalSalary());
61 }
62 }

```

```

cd C:\Users\Laptop inn\Documents\NetBeans\Projects\mavenproject2; "JAVA_HOME"
Running NetBeans Compile On Save execution. Phase execution is skipped and
Scanning for projects...

```

```

-----< com.mycompany:mavenproject2 >-----
Building mavenproject2 1.0-SNAPSHOT
-----[ jar ]-----

```

```

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject2 ---
Hello World!

```

```

BUILD SUCCESS

```

```

Total time: 1.630 s
Finished at: 2024-04-26T15:22:21+05:00

```

```

|

```

## Task 4

```
1  class Employee {
2      private String name;
3      private int age;
4      private String phoneNumber;
5      private String address;
6      private double salary;
7
8      // Constructors
9      public Employee() {
10     }
11
12     public Employee(String name, int age, String phoneNumber, String address, double salary) {
13         this.name = name;
14         this.age = age;
15         this.phoneNumber = phoneNumber;
16         this.address = address;
17         this.salary = salary;
18     }
19
20     public Employee(String name, int age) {
21         this.name = name;
22         this.age = age;
23     }
24
25     // Getter and Setter methods for all data members
26     public String getName() {
27         return name;
28     }
29
30     public void setName(String name) {
31         this.name = name;
32     }
33
34     public int getAge() {
35         return age;
36     }
37
38     public void setAge(int age) {
39         this.age = age;
40     }
41 }
```

```
40     }
41
42     public String getPhoneNumber() {
43         return phoneNumber;
44     }
45
46     public void setPhoneNumber(String phoneNumber) {
47         this.phoneNumber = phoneNumber;
48     }
49
50     public String getAddress() {
51         return address;
52     }
53
54     public void setAddress(String address) {
55         this.address = address;
56     }
57
58     public double getSalary() {
59         return salary;
60     }
61
62     public void setSalary(double salary) {
63         this.salary = salary;
64     }
65
66     // Method to print salary
67     public void printSalary() {
68         System.out.println("Salary: $" + salary);
69     }
70
71     // Method to print all information
72     public void printInfo() {
73         System.out.println("Name: " + name);
74         System.out.println("Age: " + age);
75         System.out.println("Phone Number: " + phoneNumber);
76         System.out.println("Address: " + address);
77         System.out.println("Salary: $" + salary);
78     }
79 }
```



```

80
81 class Programmer extends Employee {
82     private double bonus;
83     private String specialization;
84
85     public Programmer(String name, int age, String phoneNumber, String address, double salary,
86         double bonus, String specialization) {
87         super(name, age, phoneNumber, address, salary);
88         this.bonus = bonus;
89         this.specialization = specialization;
90     }
91
92     // Method to display Programmer's details
93     public void display() {
94         System.out.println("Name: " + getName());
95         System.out.println("Age: " + getAge());
96         System.out.println("Phone Number: " + getPhoneNumber());
97         System.out.println("Address: " + getAddress());
98         System.out.println("Salary: $" + getSalary());
99         System.out.println("Bonus: $" + bonus);
100        System.out.println("Specialization: " + specialization);
101    }
102 }
103
104 class Manager extends Employee {
105     private String department;
106
107     public Manager(String name, int age, String phoneNumber, String address, double salary,
108         String department) {
109         super(name, age, phoneNumber, address, salary);
110         this.department = department;
111     }
112
113     // Method to display Manager's details
114     public void display() {
115         System.out.println("Name: " + getName());
116         System.out.println("Age: " + getAge());
117         System.out.println("Phone Number: " + getPhoneNumber());
118         System.out.println("Address: " + getAddress());
119         System.out.println("Salary: $" + getSalary());
120         System.out.println("Department: " + department);

```

```

120         System.out.println("Department: " + department);
121     }
122 }
123
124 public class Main {
125     public static void main(String[] args) {
126         // Create an object of Programmer class
127         Programmer programmer = new Programmer("John", 30, "1234567890", "123 Main St", 5000,
128             500, "Java");
129
130         // Invoke methods accessible by Programmer object
131         programmer.printInfo();
132         programmer.printSalary();
133         programmer.display();
134     }
135 }

```

```
-----< com.mycompany:mavenproject2 >-----  
| Building mavenproject2 1.0-SNAPSHOT  
-----[ jar ]-----  
  
| --- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject2 ---  
Hello World!  
-----  
BUILD SUCCESS  
-----  
Total time: 1.618 s  
Finished at: 2024-04-26T15:27:30+05:00  
-----
```

## Task 5

```
// Superclass vehicle
class Vehicle {
    protected String brand;

    // Constructor
    public Vehicle(String brand) {
        this.brand = brand;
    }

    // Method to display vehicle information
    public void displayInfo() {
        System.out.println("Brand: " + brand);
    }
}

// Subclass Car
class Car extends Vehicle {
    private String model;

    // Constructor
    public Car(String brand, String model) {
        super(brand); // Call superclass constructor
        this.model = model;
    }

    // Method to display car information
    public void displayCarInfo() {
        super.displayInfo(); // Call superclass method
        System.out.println("Model: " + model);
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a Car object
        Car car = new Car("Toyota", "Camry");

        // Display car information
        car.displayCarInfo();
    }
}
```