

Java Programming Practical

1.a] Write a program to create a class and implement a default, overloaded and copy Constructor.

```
1  /*
2   Write a program to create a class and implement a default, overloaded and copy
3   Constructor.
4   */
5   package OOPsConceptsInJavaPart1;
6
7   class Book {
8
9       String title;
10      String author;
11
12      //Default Constructor
13      public Book() {
14          this.title = "Unknown Title";
15          this.author = "Unknown Author";
16      }
17
18      //Overloaded Constructor
19      public Book(String t, String a) {
20          this.title = t;
21          this.author = a;
22      }
23
24      //Copy Constructor
25      public Book(Book anotherBook) {
26          this.title = anotherBook.title;
27          this.author = anotherBook.author;
28      }
29
30      public void display() {
31          System.out.println("Title: " + title + ", Author: " + author);
32      }
33  }
```

```
34
35  public class Library {
36      public static void main(String[] args) {
37          Book b1 = new Book();
38          b1.display();
39
40          Book b2 = new Book("Book of Optics", "Ibn al-Haytham");
41          b2.display();
42
43          Book b3 = new Book(b2);
44          b3.display();
45      }
46  }
```

1.b] Write a program to create a class and implement the concepts of Method Overloading

```
1  /*
2  |  Write a program to create a class and implement the concepts of Method
3  |  Overloading.
4  |  */
5  |  package OOPsConceptsInJavaPart1;
6  |
7  |  class Book {
8  |
9  |      static int bookId = 0;
10 |      String title;
11 |      String author;
12 |
13 |      public Book(String t, String a) {
14 |          bookId += 1;
15 |          this.title = t;
16 |          this.author = a;
17 |      }
18 |
19 |      public void searchBook(int id) {
20 |          System.out.println("Searching for book with Id: " + id);
21 |      }
22 |
23 |      public void searchBook(String t) {
24 |          System.out.println("Searching for book with title: " + t);
25 |      }
26 |
27 |      public void searchBook(String t, String a) {
28 |          System.out.println("Searching for book with title: " + t + " and Author: " + a);
29 |      }
30 |
31 |  }
32 |
33 |  public class Library {
34 |
35 |      public static void main(String[] args) {
36 |          Book b = new Book("Book of Optics", "Ibn al-Haytham");
37 |          b.searchBook(1);
38 |          b.searchBook("Book of Optics");
39 |          b.searchBook("Book of Optics", "Ibn al-Haytham");
40 |      }
41 |  }
```

1.c] Write a program to create a class and implement the concepts of Static methods.

```
1  /*
2  Write a program to create a class and implement the concepts of Static methods
3  */
4  package OOPsConceptsInJavaPart1;
5
6  class Book {
7
8      String title;
9      String author;
10     static int totalBooks = 0;
11
12     //Overloaded Constructor
13     public Book(String t, String a) {
14         this.title = t;
15         this.author = a;
16         totalBooks++;
17     }
18
19     public void display() {
20         System.out.println("Title: " + title + ", Author: " + author);
21     }
22
23     public static void showLibraryInfo() {
24         System.out.println("Welcome to the Library System!");
25         System.out.println("Total Books Available: " + totalBooks);
26     }
27 }
28
29 public class Library {
30
31     public static void main(String[] args) {
32         Book b1 = new Book("Book of Optics", "Ibn al-Haytham");
33         b1.display();
34
35         Book b2 = new Book("The Canon of Medicine", "Ibn Sina");
36         b2.display();
37
38         Book b3 = new Book("Al-Jabr (Algebra)", "Al Khawarizmi");
39         b3.display();
40         Book.showLibraryInfo();
41     }
42 }
43 }
```

2.a] Write a program to implement the concepts of Inheritance and Method overriding.

```
1  /*
2  Write a program to implement the concepts of Inheritance and Method overriding
3  */
4  package OOPsConceptsInJavaPart2;
5
6  class Vehicle {
7
8      String brand;
9      int speed;
10
11     public Vehicle(String b, int s) {
12         brand = b;
13         speed = s;
14     }
15
16     public void displayInfo() {
17         System.out.println("Vehicle Brand: " + brand + ", Speed: " + speed + " km/h");
18     }
19 }
20
21 class Bike extends Vehicle {
22
23     public Bike(String b, int s) {
24         super(b, s);
25     }
26
27     @Override
28     public void displayInfo() {
29         System.out.println("Bike Brand: " + brand + ", Speed: " + speed + " km/h");
30     }
31 }
32
33 class Car extends Vehicle {
34
35     public Car(String b, int s) {
36         super(b, s);
37     }
38
39     @Override
40     public void displayInfo() {
41         System.out.println("Car Brand: " + brand + ", Speed: " + speed + " km/h");
42     }
43 }
```

```

44
45 public class VehicleSystem {
46
47     public static void main(String[] args) {
48         Vehicle vehicle = new Vehicle("Vehicle Brand", 80);
49         Vehicle bike = new Bike("Yamaha RX100", 100);
50         Vehicle car = new Car("Mahindra Thar", 120);
51
52         vehicle.displayInfo();
53         bike.displayInfo();
54         car.displayInfo();
55     }
56 }

```

2.b] Write a program to implement the concepts of Abstract classes and methods.

```

1  /*
2   Write a program to implement the concepts of Abstract classes and methods
3   */
4  package OOPsConceptsInJavaPart2;
5
6  abstract class Employee {
7      String name;
8      int empId;
9
10     // Constructor
11     public Employee(String name, int empId) {
12         this.name = name;
13         this.empId = empId;
14     }
15
16     // Abstract Method (Must be implemented by subclasses)
17     abstract double calculateSalary();
18
19     // Concrete Method
20     public void displayDetails() {
21         System.out.println("Employee ID: " + empId + ", Name: " + name);
22     }
23 }

```

```

24
25 // Subclass 1: Full-Time Employee
26 class FullTimeEmployee extends Employee {
27     double monthlySalary;
28
29     public FullTimeEmployee(String name, int empId, double salary) {
30         super(name, empId);
31         this.monthlySalary = salary;
32     }
33
34     // Implement abstract method
35     @Override
36     double calculateSalary() {
37         return monthlySalary;
38     }
39 }
40
41 // Subclass 2: Part-Time Employee
42 class PartTimeEmployee extends Employee {
43     double hourlyRate;
44     int hoursWorked;
45
46     public PartTimeEmployee(String name, int empId, double hourlyRate, int hoursWorked) {
47         super(name, empId);
48         this.hourlyRate = hourlyRate;
49         this.hoursWorked = hoursWorked;
50     }
51
52     // Implement abstract method
53     @Override
54     double calculateSalary() {
55         return hourlyRate * hoursWorked;
56     }
57 }
58
59 // Main Class
60 public class EmployeeManagement {
61     public static void main(String[] args) {
62         Employee emp1 = new FullTimeEmployee("Alice", 101, 50000);
63         Employee emp2 = new PartTimeEmployee("Bob", 102, 500, 20);
64
65         emp1.displayDetails();
66         System.out.println("Salary: " + emp1.calculateSalary());
67
68         emp2.displayDetails();
69         System.out.println("Salary: " + emp2.calculateSalary());
70     }
71 }

```

2.c] Write a program to implement the concept of interfaces.

```
1  /*
2   Write a program to implement the concept of interfaces.
3   */
4   package OOPsConceptsInJavaPart2;
5
6   // Interface defining payment behavior
7   interface Payment {
8       void makePayment(double amount);
9   }
10
11  // Class implementing Payment using Credit Card
12  class CreditCardPayment implements Payment {
13      private String cardNumber;
14
15      public CreditCardPayment(String cardNumber) {
16          this.cardNumber = cardNumber;
17      }
18
19      @Override
20      public void makePayment(double amount) {
21          System.out.println("Paid ₹" + amount + " using Credit Card: " + cardNumber);
22      }
23  }
24
25  // Class implementing Payment using UPI
26  class UPIPayment implements Payment {
27      private String upiId;
28
29      public UPIPayment(String upiId) {
30          this.upiId = upiId;
31      }
32
33      @Override
34      public void makePayment(double amount) {
35          System.out.println("Paid ₹" + amount + " using UPI ID: " + upiId);
36      }
37  }
38
39  // Main Class
40  public class PaymentSystem {
41      public static void main(String[] args) {
42          Payment payment1 = new CreditCardPayment("1234-5678-9876-5432");
43          Payment payment2 = new UPIPayment("user@upi");
44
45          payment1.makePayment(1500);
46          payment2.makePayment(800);
47      }
48  }
```

7.a] Flow Layout.

```
1 package Layouts;
2
3
4 import java.awt.FlowLayout;
5 import javax.swing.JButton;
6 import javax.swing.JFrame;
7
8 public class FlowLayoutExample {
9     public static void main(String[] args) {
10         JFrame frame = new JFrame("Flow Layout Example");
11
12         FlowLayout layout = new FlowLayout();
13         frame.setLayout(layout);
14
15         for(int i=0;i<5;i++){
16             JButton button = new JButton("My Button "+i);
17             frame.add(button);
18         }
19
20         frame.setSize(500,500);
21         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22         frame.setVisible(true);
23     }
24 }
```

7.b] Grid Layout

```
1 package Layouts;
2
3 import java.awt.GridLayout;
4 import javax.swing.JButton;
5 import javax.swing.JFrame;
6
7 public class GridLayoutExample {
8     public static void main(String[] args) {
9         JFrame frame = new JFrame("Flow Layout Example");
10
11         GridLayout layout = new GridLayout(3,4);
12         frame.setLayout(layout);
13
14         for(int i=0;i<12;i++){
15             JButton button = new JButton("My Button "+i);
16             frame.add(button);
17         }
18
19         frame.setSize(500,500);
20         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21         frame.setVisible(true);
22     }
23 }
```


7.c] Border Layout

```
1  package Layouts;
2
3  import java.awt.BorderLayout;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6
7  public class BorderLayoutExample {
8
9      public static void main(String[] args) {
10         JFrame frame = new JFrame("Flow Layout Example");
11
12         BorderLayout layout = new BorderLayout();
13         frame.setLayout(layout);
14
15         JButton nButton = new JButton("North");
16         JButton sButton = new JButton("South");
17         JButton wButton = new JButton("West");
18         JButton eButton = new JButton("East");
19         JButton cButton = new JButton("Center");
20
21         frame.add(nButton, BorderLayout.NORTH);
22         frame.add(sButton, BorderLayout.SOUTH);
23         frame.add(wButton, BorderLayout.WEST);
24         frame.add(eButton, BorderLayout.EAST);
25         frame.add(cButton, BorderLayout.CENTER);
26
27         frame.setSize(500, 500);
28         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29         frame.setVisible(true);
30     }
31 }
```

8.a] Action Event

```
1  package EventHandling;
2
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8
9  public class ActionEventExample {
10
11     public static void main(String[] args) {
12         JFrame frame = new JFrame("Action Event Example");
13
14         JButton button = new JButton("My Button");
15         frame.add(button);
16
17         ButtonActionListner listner = new ButtonActionListner(frame);
18         button.addActionListener(listner);
19
20         frame.setSize(500, 500);
21         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22         frame.setVisible(true);
23     }
24 }
25
26 class ButtonActionListner implements ActionListener {
27
28     private JFrame parentFrame;
29
30     public ButtonActionListner(JFrame frame) {
31         this.parentFrame = frame;
32     }
33
34     @Override
35     public void actionPerformed(ActionEvent e) {
36         JOptionPane.showMessageDialog(parentFrame, e.getActionCommand() + " clicked");
37     }
38
39 }
```

8.b] Mouse Event

Note: Same practical can be used for practical 10.

```
1  package EventHandling;
2
3  import java.awt.Font;
4  import java.awt.event.MouseEvent;
5  import java.awt.event.MouseListener;
6  import javax.swing.JFrame;
7  import javax.swing.JLabel;
8
9  public class MouseEventExample {
10
11     public static void main(String[] args) {
12         JFrame frame = new JFrame("Mouse Event Example");
13
14         JLabel label = new JLabel("Hello", JLabel.CENTER);
15         label.setFont(new Font("Arial", Font.ITALIC, 20));
16         frame.add(label);
17
18         label.addMouseListener(new MouseListener() {
19
20             public void mouseExited(MouseEvent e) {
21                 System.out.println(e.getX() + "," + e.getY());
22             }
23
24             public void mouseEntered(MouseEvent e) {
25                 System.out.println(e.getX() + "," + e.getY());
26             }
27
28             public void mouseReleased(MouseEvent e) {
29
30
31
32
33                 public void mousePressed(MouseEvent e) {
34
35
36
37                     public void mouseClicked(MouseEvent e) {
38                         System.out.println(e.getButton());
39                     }
40                 });
41
42                 frame.setSize(500, 500);
43                 frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
44                 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45                 frame.setVisible(true);
46             }
47         });
48     }
49 }
```

8.c] Key Event

Note: Same practical can be used for practical 9.

```
1  package EventHandlering;
2
3  import java.awt.event.KeyAdapter;
4  import java.awt.event.KeyEvent;
5  import javax.swing.JFrame;
6  import javax.swing.JOptionPane;
7
8  public class KeyEventExample {
9
10     public static void main(String[] args) {
11         JFrame frame = new JFrame("Key Event Example");
12
13         frame.addKeyListener(new KeyAdapter() {
14             @Override
15             public void keyPressed(KeyEvent e) {
16                 JOptionPane.showMessageDialog(frame, e.getKeyChar());
17             }
18         });
19
20         frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
21         frame.setSize(450, 600);
22         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23         frame.setVisible(true);
24     }
25 }
```