by Kunal Sir

Assignment : Method Overloading (Compile-Time Polymorphism)

Part A: MCQs on Method Overloading

Q1. What is method overloading in Java?

- A. Same method name with same parameters in different classes
- B. Same method name with different parameter lists in the same class
- C. Different method names with same parameters
- D. Same method name and parameters with different return types

☞ **Answer: B** – Same method name with different parameter lists in the same class

---

Q2. Which of the following changes results in valid method overloading?

- A. Changing only return type
- B. Changing method name
- C. Changing number of parameters
- D. Changing access modifier only

☞ **Answer: C** – Changing number of parameters

Q3. Method overloading is an example of:

- A. Runtime polymorphism
- B. Compile-time polymorphism
- C. Dynamic binding
- D. Method overriding

☞ **Answer: B** – Compile-time polymorphism

Q4. Which of the following method signatures are overloaded?

```
void display(int a)
void display(double a)
```

- A. Invalid (same signature)
- B. Valid overloading
- C. Causes runtime error
- D. Causes ambiguity

☞ **Answer: B** – Valid overloading

Q5. Can constructors be overloaded in Java?

- A. No, never
- B. Yes, like methods
- C. Only default constructors
- D. Only parameterized constructors

☞ **Answer: B** – Yes, like methods

Q6. Which rule is mandatory for method overloading?

- A. Same return type
- B. Same access modifier
- C. Same method name
- D. Same method body

☞ **Answer: C** – Same method name

Q7. What happens if two overloaded methods differ only by return type?

- A. Code compiles successfully
- B. JVM selects correct method
- C. Compile-time error
- D. Runtime exception

☞ **Answer: C** – Compile-time error

Q8. Which of the following is NOT used by the compiler to resolve overloading?

- A. Method name
- B. Number of parameters
- C. Data type of parameters
- D. Return type

☞ **Answer: D** – Return type

Q9. Method overloading improves:

- A. Security
- B. Performance
- C. Code readability
- D. Memory usage

☞ **Answer: C** – Code readability

Q10. Which is a correct example of method overloading?

- A. `int sum(int a, int b)` and `double sum(int a, int b)`
- B. `void sum(int a)` and `void sum(int a, int b)`
- C. `void sum(int a)` and `int sum(int a)`
- D. `static void main()` and `static int main()`

☞ **Answer: B** – `void sum(int a)` and `void sum(int a, int b)`

Part B: Problem Statements on Method Overloading (with Sample Input & Output)

Problem 1: Create a class `Calculator` with overloaded methods `add()`.

Sample Input:

- add(10, 20)
- add(5, 10, 15)
- add(2.5, 3.5)

Sample Output:

- Sum = 30
- Sum = 30
- Sum = 6.0

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
```

```
        return a + b + c;
    }

    double add(double a, double b) {
        return a + b;
    }
}
```

Problem 2: Create a class `Shape` with overloaded method `area()`.

Sample Input:

- area(7) // Circle
- area(5, 4) // Rectangle
- area(6, 3, true) // Triangle

Sample Output:

- Area of Circle = 153.94
- Area of Rectangle = 20
- Area of Triangle = 9


```
class Shape {
    double area(int r) {
        return 3.14 * r * r;
    }

    int area(int l, int b) {
        return l * b;
    }

    double area(int b, int h, boolean triangle) {
        return 0.5 * b * h;
    }
}
```

Problem 3: Create a class `Printer` with overloaded method `print()`.

Sample Input:

- print(100)
- print("Java")
- print(10, "Programs")

Sample Output:

- 100
- Java
- 10 Programs


```
class Printer {
    void print(int a) {
        System.out.println(a);
    }

    void print(String s) {
        System.out.println(s);
    }

    void print(int a, String s) {
        System.out.println(a + " " + s);
    }
}
```

Problem 4: Create a class `MathOperation` with overloaded method `multiply()`.

Sample Input:

- multiply(2, 3)
- multiply(2, 3, 4)
- multiply(2.5f, 4.0f)

Sample Output:

- Result = 6
- Result = 24
- Result = 10.0


```
class MathOperation {
    int multiply(int a, int b) {
        return a * b;
    }

    int multiply(int a, int b, int c) {
        return a * b * c;
    }

    float multiply(float a, float b) {
        return a * b;
    }
```

```
}
```

Problem 5: Create a class `Display` with overloaded method `show()`.

Sample Input:

- show("Amit")
- show("Amit", 20)
- show("Amit", 20, 85.5)

Sample Output:

- Name: Amit
- Name: Amit, Age: 20
- Name: Amit, Age: 20, Marks: 85.5


```java
class Display {
    void show(String name) {
        System.out.println("Name: " + name);
    }

    void show(String name, int age) {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    void show(String name, int age, double marks) {
        System.out.println("Name: " + name + ", Age: " + age + ", Marks: " + marks);
    }
}
```

Problem 6: Create a class `Converter` with overloaded method `convert()`.

Sample Input:

- convert(5)
- convert(200, true)

Sample Output:

- 5000 meters
- 20000 centimeters

```
class Converter {
    void convert(int km) {
        System.out.println(km * 1000 + " meters");
    }

    void convert(int meters, boolean flag) {
        System.out.println(meters * 100 + " centimeters");
    }
}
```

Problem 7: Create a class `Salary` with overloaded method `calculateSalary()`.

Sample Input:

- calculateSalary(20000)
- calculateSalary(20000, 5000)

Sample Output:

- Total Salary = 20000
- Total Salary = 25000


```
class Salary {
    int calculateSalary(int basic) {
        return basic;
    }

    int calculateSalary(int basic, int bonus) {
        return basic + bonus;
    }
}
```

Problem 8: Create a class `AreaCalculator` with overloaded method `calculate()`.

Sample Input:

- calculate(4)
- calculate(5, 6)

Sample Output:

- Area of Square = 16
- Area of Rectangle = 30

```
class AreaCalculator {
    int calculate(int side) {
        return side * side;
    }

    int calculate(int l, int b) {
        return l * b;
    }
}
```

Problem 9: Create a class `Message` with overloaded method `send()`.

Sample Input:

- send("Hello")
- send("Meeting Today", "High")

Sample Output:

- Message Sent: Hello
- Message Sent: Meeting Today | Priority: High

```
class Message {
    void send(String msg) {
        System.out.println("Message Sent: " + msg);
    }

    void send(String msg, String priority) {
        System.out.println("Message Sent: " + msg + " | Priority: " + priority);
    }
}
```

Problem 10: Create a class `Result` with overloaded method `grade()`.

Sample Input:

- grade(85)
- grade(85, 90)

Sample Output:

- Grade: A
- Grade: A (Attendance Eligible)

```
class Result {
    void grade(int marks) {
        System.out.println("Grade: A");
    }

    void grade(int marks, int attendance) {
        System.out.println("Grade: A (Attendance Eligible)");
    }
}
```

Part C: Problem Statements on Constructor Overloading (with Sample Input & Output)

Problem 1: `Student` class with multiple constructors.

Sample Input:

- new Student()
- new Student("Rahul")
- new Student("Rahul", 101)

Sample Output:

- Student Created
- Name: Rahul
- Name: Rahul, Roll No: 101

```
class Student {
    Student() {
        System.out.println("Student Created");
    }

    Student(String name) {
        System.out.println("Name: " + name);
    }

    Student(String name, int roll) {
        System.out.println("Name: " + name + ", Roll No: " + roll);
    }
}
```

Problem 2: `Employee` class with overloaded constructors.

Sample Input:

- new Employee(1)
- new Employee(1, "Neha")
- new Employee(1, "Neha", 45000)

Sample Output:

- ID: 1
- ID: 1, Name: Neha
- ID: 1, Name: Neha, Salary: 45000

```
class Employee {
    Employee(int id) {
        System.out.println("ID: " + id);
    }

    Employee(int id, String name) {
        System.out.println("ID: " + id + ", Name: " + name);
    }

    Employee(int id, String name, int salary) {
        System.out.println("ID: " + id + ", Name: " + name + ", Salary: " + salary);
    }
}
```

Problem 3: `Book` class constructor overloading.

Sample Input:

- new Book("Java Basics")
- new Book("Java Basics", "James")
- new Book("Java Basics", "James", 499)

Sample Output:

- Title: Java Basics
- Title: Java Basics, Author: James
- Title: Java Basics, Author: James, Price: 499

```
class Book {
    Book(String title) {
        System.out.println("Title: " + title);
    }

    Book(String title, String author) {
        System.out.println("Title: " + title + ", Author: " + author);
    }

    Book(String title, String author, int price) {
        System.out.println("Title: " + title + ", Author: " + author + ", Price: " +
price);
    }
}
```

Problem 4: `Rectangle` class with overloaded constructors.

Sample Input:

- new Rectangle()
- new Rectangle(5, 10)

Sample Output:

- Area = 0
- Area = 50

```
class Rectangle {
    Rectangle() {
        System.out.println("Area = 0");
    }

    Rectangle(int l, int b) {
        System.out.println("Area = " + (l * b));
    }
}
```

Problem 5: `Circle` class constructor overloading.

Sample Input:

- new Circle()
- new Circle(7)

Sample Output:

- Radius = 0
- Area = 153.94

```
class Circle {
    Circle() {
        System.out.println("Radius = 0");
    }

    Circle(int r) {
        System.out.println("Area = " + (3.14 * r * r));
    }
}
```

Problem 6: `Car` class constructor overloading.

Sample Input:

- new Car("Toyota")
- new Car("Toyota", "Innova")
- new Car("Toyota", "Innova", 2500000)

Sample Output:

- Brand: Toyota
- Brand: Toyota, Model: Innova
- Brand: Toyota, Model: Innova, Price: 2500000

```
class Car {
    Car(String brand) {
        System.out.println("Brand: " + brand);
    }

    Car(String brand, String model) {
        System.out.println("Brand: " + brand + ", Model: " + model);
    }

    Car(String brand, String model, int price) {
        System.out.println("Brand: " + brand + ", Model: " + model + ", Price: " +
price);
    }
}
```

Problem 7: `BankAccount` class constructor overloading.

Sample Input:

- new BankAccount(12345)
- new BankAccount(12345, "Ravi")
- new BankAccount(12345, "Ravi", 50000)

Sample Output:

- Account No: 12345
- Account No: 12345, Name: Ravi
- Account No: 12345, Name: Ravi, Balance: 50000


```
class BankAccount {
    BankAccount(int accNo) {
        System.out.println("Account No: " + accNo);
    }

    BankAccount(int accNo, String name) {
        System.out.println("Account No: " + accNo + ", Name: " + name);
    }

    BankAccount(int accNo, String name, int balance) {
        System.out.println("Account No: " + accNo + ", Name: " + name + ", Balance: "
+ balance);
    }
}
```

Problem 8: `Laptop` class constructor overloading.

Sample Input:

- new Laptop("Dell")
- new Laptop("Dell", 8)
- new Laptop("Dell", 8, 60000)

Sample Output:

- Brand: Dell
- Brand: Dell, RAM: 8GB
- Brand: Dell, RAM: 8GB, Price: 60000

```
class Laptop {
    Laptop(String brand) {
        System.out.println("Brand: " + brand);
    }

    Laptop(String brand, int ram) {
        System.out.println("Brand: " + brand + ", RAM: " + ram + "GB");
    }

    Laptop(String brand, int ram, int price) {
        System.out.println("Brand: " + brand + ", RAM: " + ram + "GB, Price: " +
price);
    }
}
```

Problem 9: `Product` class constructor overloading.

Sample Input:

- new Product("Pen")
- new Product("Pen", 10)
- new Product("Pen", 10, 100)

Sample Output:

- Product: Pen
- Product: Pen, Quantity: 10
- Product: Pen, Quantity: 10, Price: 100

```
class Product {
    Product(String name) {
        System.out.println("Product: " + name);
    }

    Product(String name, int qty) {
        System.out.println("Product: " + name + ", Quantity: " + qty);
    }

    Product(String name, int qty, int price) {
        System.out.println("Product: " + name + ", Quantity: " + qty + ", Price: " +
price);
    }
}
```

Problem 10: `Person` class constructor overloading.

Sample Input:

- new Person("Anita")
- new Person("Anita", 22)
- new Person("Anita", 22, "Pune")

Sample Output:

- Name: Anita
- Name: Anita, Age: 22
- Name: Anita, Age: 22, Address: Pune

```
class Person {
    Person(String name) {
        System.out.println("Name: " + name);
    }

    Person(String name, int age) {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    Person(String name, int age, String address) {
        System.out.println("Name: " + name + ", Age: " + age + ", Address: " +
address);
    }
}
```

---

by Kunal Sir

Stop, Near, 1st Floor, Above Rupam Sweets/ Priyanka Collections Building Vikas Mitra Mandal Chowk Road, Karve Nagar, Pune, Maharashtra 411052 , Mobile No.- 8888022204