Day 13

Java OOP Concepts - Polymorphism

Polymorphism

- **Polymorphism** in Java is one of the OOP concepts.
- The term "**polymorphism**" is derived from the Greek words "**poly**," meaning many, and "**morph**," meaning forms.

There are two types of polymorphism in Java

1. **Compile time polymorphism (or) method overloading (or) static banding**
2. Runtime polymorphism (or) method overriding (or) dynamic binding

Overloading in Java

- Method Overloading
- Constructor Overloading

Overloading in Java is a feature that allows a class to have **more than one method** or **constructor with the same name but different parameters**.

The method or constructor is selected based on the number and type of arguments passed during the call.

Overloading is a type of compile-time polymorphism, where the decision about which method or constructor to call is made at compile time.

Rules for Overloading (Applicable for both methods and constructors)

1. Must have a different number of parameters.
2. If the number of parameters is the same, their types must be different.
3. If the types and number of parameters are the same, their order must be different.
4. The return type of the method can be different (**It is not considered for method overloading)**.

Overloading the public static void main(String[] args) Method in Java

We can overload the main method in Java.

However, the JVM (Java Virtual Machine) specifically looks for the main method with the signature public static void main(String[] args) to start the program.

While you can create other main methods with different parameters, they will act like regular methods and won't be used as the program's starting point.

Calling the public static void main(String[] args) Method

We can call the main method from within itself by passing a String array as the argument.

This is sometimes done in recursive or testing scenarios. However, be careful with recursion to avoid creating infinite loops.

Passing Arguments to the public static void main(String[] args) Method

We can pass arguments to the main method in two ways:

1. **Using command-line arguments**: Pass the arguments when running the program from the terminal or command prompt.
2. **Using Eclipse**: Go to Run As → Run Configurations → Arguments to pass the arguments directly within the Eclipse IDE.

Quiz: Overloading Concept in Java

Question 1:

What is method overloading in Java?
A) Defining multiple methods with the same name but different return types.
B) Defining multiple methods with the same name but different parameters.
C) Defining multiple classes with the same name.
D) Defining a method inside another method.

Answer:
B) Defining multiple methods with the same name but different parameters.

Question 2:

Can we overload methods by changing only the return type of the method?
A) Yes
B) No

Answer:
B) No

Question 3:

Which of the following is a valid reason for method overloading?
A) To perform different tasks with methods that share the same name.
B) To reduce the memory footprint of the program.
C) To allow methods to be called with different return types.
D) To improve the execution speed of the program.

Answer:
A) To perform different tasks with methods that share the same name.

Question 4:

Can we overload a static method in Java?
A) Yes
B) No

Answer:
A) Yes

Question 5:

Which of the following pairs of methods would be considered overloaded methods?
A) void display(int a) and void display(int a)
B) int display(int a) and void display(int a)
C) void display(int a) and void display(double a)
D) void display(int a) and int display(double a)

Answer:
C) void display(int a) and void display(double a)

Question 6:

Which statement is true about constructor overloading?
A) You cannot overload constructors in Java.
B) Constructors can only be overloaded by changing their return types.
C) Constructors can be overloaded by changing the number and/or type of parameters.
D) Constructors can only be overloaded by changing their access modifiers.

Answer:
C) Constructors can be overloaded by changing the number and/or type of parameters.

Question 7:

Consider the following methods:

java

Copy code

```
void test(int a, double b) {}

void test(double b, int a) {}
```

Are these methods overloaded?
A) Yes
B) No

Answer:
A) Yes

Question 8:

Is it possible to overload the main method in Java?
A) Yes, and all overloaded main methods will be called by the JVM.
B) Yes, but only the main(String[] args) method will be called by the JVM.
C) No, the main method cannot be overloaded.
D) Yes, and the JVM will decide which main method to call based on the arguments.

Answer:
B) Yes, but only the main(String[] args) method will be called by the JVM.

Question 9:

Which of the following scenarios does NOT allow method overloading?
A) Changing the number of parameters.
B) Changing the data types of parameters.
C) Changing the order of parameters.
D) Changing the method's access modifier.

Answer:
D) Changing the method's access modifier.

Question 10:

What happens if you overload a method with the same parameters but a different return type?
A) The code will compile and work correctly.
B) The code will compile, but only one of the methods will be accessible.
C) The code will not compile.
D) The JVM will select the method based on the return type.

Answer:
C) The code will not compile.