| Experiment No. 6 |
| --- |
| Constructor Overloading. |
| Date of Performance:02/08/24 |
| Date of Submission:09/08/24 |
|  |

**Aim :-**Implementation of constructor overloading in object-oriented programming

**Objective :-** Implement a java program for a library management system that manages books. The system should include a Book class with attributes such as title, author, isbn, and publishedYear. Implement constructor overloading to allow the creation of Book objects in different ways. One constructor should allow the creation of a Book object with only the title and author, while another constructor should allow the creation of a Book object with title, author, and isbn, and a third constructor should allow the creation of a Book object with all four attributes: title, author, isbn, and publishedYear.

**Theory :-**

Constructor Overloading in Java is a concept in which a class can have more than one constructor with different parameter lists. Each constructor must have a unique parameter list, which differentiates it from other constructors in the class. This allows the creation of objects in different ways, providing flexibility in object initialization.

Constructor Definition:

A constructor is a special method that is called when an object is instantiated. It has the same name as the class and does not have a return type, not even void.

Parameter List:

To overload constructors, the parameter list must differ in terms of:

Number of parameters

Type of parameters

Order of parameters (if they are of different types)

Default Constructor:

If no constructor is defined, Java provides a default no-argument constructor. If any constructor is defined, the default constructor is not provided automatically.

Constructor Invocation:

Constructors can be invoked directly during the creation of an object using the new keyword. Within a class, one constructor can call another overloaded constructor using the this keyword.

Benefits of Constructor Overloading

Object Initialization Flexibility: Allows objects to be initialized in different ways based on the arguments provided.

Enhanced Readability: Improves code readability by allowing the creation of objects with different initial states.

Convenience: Provides multiple ways to instantiate a class, making it easier to use the class in various scenarios.

Syntax:  
class ClassName {

// No-argument constructor

ClassName() {

// Constructor body

}

// Constructor with one parameter

ClassName(dataType1 param1) {

// Constructor body

}

// Constructor with two parameters

ClassName(dataType1 param1, dataType2 param2) {

// Constructor body

}

// Constructor with different order of parameters

ClassName(dataType2 param2, dataType1 param1) {

// Constructor body

}

}

**Code :-**

class Book {

String title;

String author;

int isbn;

int publishedYear;

Book() {

title = "Lord Of The Rings";

author = "Tolkien";

isbn = 100123;

publishedYear = 1976;

}

Book(String bookName, String bookAuthor) {

title = bookName;

author = bookAuthor;

isbn = 0;

publishedYear = 0;

System.out.println("Constructor with two arguments: " + title + " by " + author);

}

void displayInfo() {

System.out.println("Title: " + title);

System.out.println("Author: " + author);

System.out.println("ISBN: " + isbn);

System.out.println("Published Year: " + publishedYear);

}

public static void main(String[] args) {

Book b1 = new Book();

System.out.println("Details of Book 1:");

b1.displayInfo();

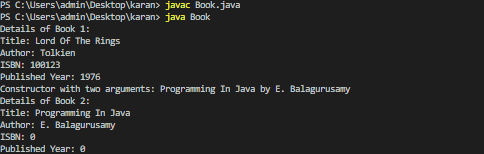
Book b2 = new Book("Programming In Java", "E. Balagurusamy");

System.out.println("Details of Book 2:");

b2.displayInfo();

}}

**OUTPUT:-**

****

**Conclusion :-** This experiment effectively demonstrates constructor overloading in Java, illustrating how multiple constructors allow for the creation of objects with varying initial values. By providing both a no-argument constructor and a parameterized constructor, it enhances flexibility and code organization, showcasing a key principle of object-oriented programming that facilitates better data management.