# **National University of Computer & Emerging Sciences, Karachi**

**Software Engineering Department**

**Spring 2024, Lab Manual – 04**

|  |  |
| --- | --- |
| **Course Code: CL-217** | **Course: Object Oriented Programming Lab** |
| **Instructor(s):** | **Ali Fatmi** |

LAB - 4

Constructors and Garbage Collection

**Constructor**

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the ***new*** keyword, at least one constructor is called.

Example: Calling constructor for creating Box class object



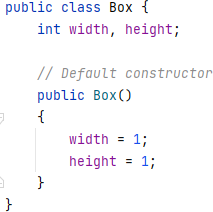
# Types of Constructors

There are two types of constructors in Java: default (no parameter) constructor, and parameterized constructor.

## Default Constructor

The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

* Example of Default (no parameter) Constructor

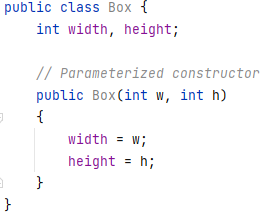


## Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.

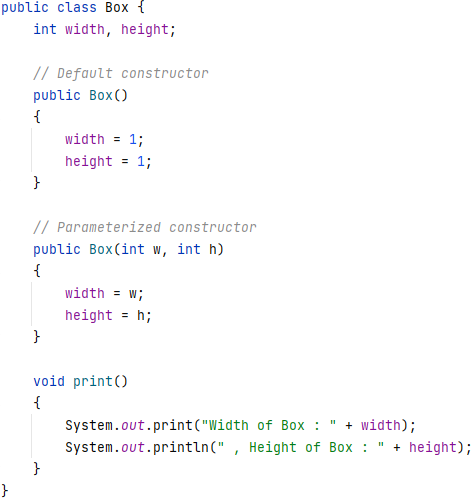
The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

* Example of Parameterized Constructor

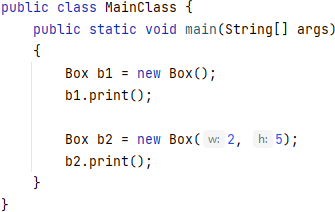


# Rules for Constructors

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. If there is no constructor in a class, compiler automatically creates a default constructor. Complete Example



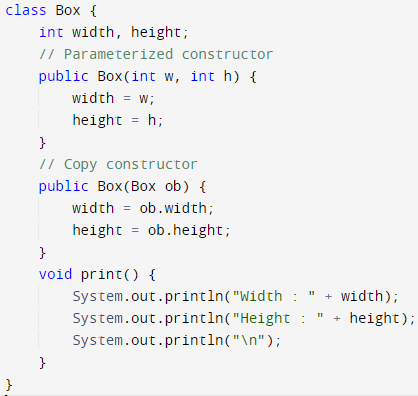
*// Using these constructors from main class:*

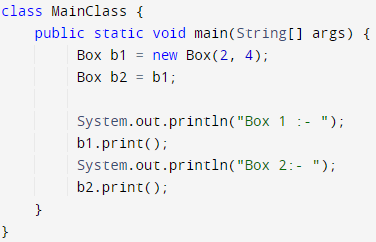


# Copy Constructor

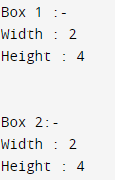
We can use one object to initialize another object in Java by passing the object as parameter to a constructor.

Example:





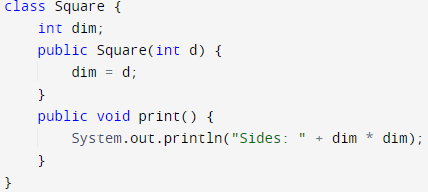
Output:

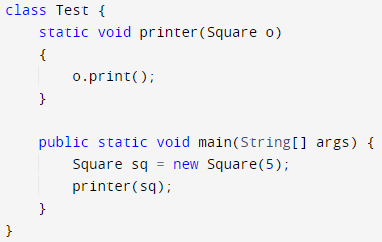


# Object Passing using Function

We can also pass objects as parameters to functions in Java just like other primitive type arguments.

Example:





Output:



# Garbage Collection

In Java, when we create an object of the class it occupies some space in the memory (heap). If we do not delete these objects, it remains in the memory and occupies unnecessary space that is not upright from the aspect of programming. To resolve this problem, we use the **Destructor**.

Unlike C++, which has explicit destructor, Java uses **finalize( )** method for this purpose. Since it cannot be predicted when the Java garbage collection occurs.

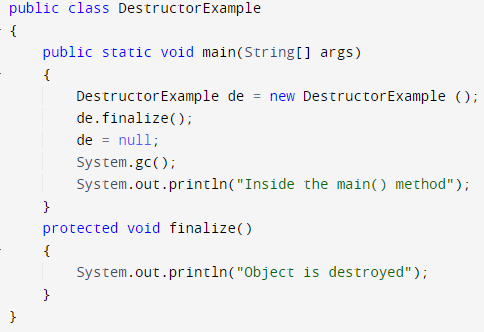
# The finalize( ) Method

It is difficult for the programmer to forcefully execute the garbage collector to destroy the object. But

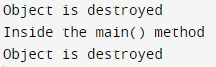
Java provides an alternative way to do the same. The Java Object class provides the finalize() method that works the same as the destructor.

* It is a protected method of Object class
* It can be called only once.
* We need to call the finalize() method explicitly if we want to override the method.
* The gc() is a method of JVM executed by the Garbage Collector. It invokes when the heap memory is full and requires more memory for new arriving objects.

Example



Output:



Lab Tasks

1. Employee Management System:

Scenario: In a multinational company, there's a need to manage employee information efficiently. Design an Employee Management System where each employee's details such as ID, name, designation, and salary are stored. Implement constructors to handle scenarios like new employee hiring, employee data retrieval, and employee transfer between departments. Ensure proper garbage collection to release memory when an employee leaves the company.

Attributes:

Employee: employeeId, name, designation, salary

Functions:

Employee(): Default constructor

Employee(int employeeId, String name, String designation, double salary): Parameterized constructor

Employee(Employee otherEmployee): Copy constructor

displayDetails(): Display employee details

1. Inventory Management System:

Scenario: A retail chain requires an Inventory Management System to keep track of products in stock. Create a system where each product's information, including ID, name, price, and quantity available, is stored. Implement constructors to manage scenarios like adding new products, updating existing product details, and restocking inventory. Use garbage collection to handle product objects that are no longer needed.

Attributes:

Product: productId, name, price, quantityAvailable

Functions:

Product(): Default constructor

Product(int productId, String name, double price, int quantityAvailable): Parameterized constructor

Product(Product otherProduct): Copy constructor

updateQuantity(int quantity): Update product quantity

displayProductDetails(): Display product details

1. Library Management System:

Scenario: A library needs a software system to automate its book management tasks. Design a Library Management System where each book's details like ID, title, author, and availability status are stored. Implement constructors to manage scenarios like adding new books to the library, checking out books to borrowers, and returning books to the library. Utilize garbage collection to remove book objects when they are no longer in circulation.

**Attributes:**

Book: bookId, title, author, availability

**Functions:**

Book(): Default constructor

Book(int bookId, String title, String author, boolean availability): Parameterized constructor

Book(Book otherBook): Copy constructor

checkoutBook(): Check out a book

returnBook(): Return a book

displayBookDetails(): Display book details

1. **Vehicle Rental System:**

Scenario: A car rental company wants to streamline its vehicle rental process. Develop a Vehicle Rental System where each vehicle's information, including ID, type, brand, and rental rate, is maintained. Implement constructors to handle scenarios like adding new vehicles to the rental fleet, renting out vehicles to customers, and receiving returned vehicles. Ensure proper garbage collection to manage vehicle objects efficiently.

**Attributes:**

Vehicle: vehicleId, type, brand, rentalRate

**Functions:**

Vehicle(): Default constructor

Vehicle(int vehicleId, String type, String brand, double rentalRate): Parameterized constructor

Vehicle(Vehicle otherVehicle): Copy constructor

rentVehicle(): Rent a vehicle

returnVehicle(): Return a vehicle

displayVehicleDetails(): Display vehicle details

1. **Banking System:**

Scenario: A bank aims to enhance its banking services by implementing a robust banking system. Create a Banking System where each account's details such as ID, type, balance, and customer information are stored. Implement constructors to manage scenarios like opening new accounts, processing transactions (deposits, withdrawals), and generating account statements. Utilize garbage collection to free up memory used by closed accounts or inactive customers.

**Attributes:**

Account: accountId, accountType, balance, customer

**Functions:**

Account(): Default constructor

Account(int accountId, String accountType, double balance, Customer customer): Parameterized constructor

Account(Account otherAccount): Copy constructor

deposit(double amount): Deposit funds

withdraw(double amount): Withdraw funds

displayAccountInfo(): Display account information