

**Constructors, Java Keywords - Assignment Problem (Any 4) Problem 1: Movie Ticket Booking** ��

Design a **Movie Ticket System**.

● Class MovieTicket with fields: String movieName, String theatreName, int seatNumber, double price.

● Implement constructor overloading:

1. Default constructor → assigns "Unknown" movie.

2. Constructor with movie name → assigns default price = 200.

3. Constructor with movie name and seat number → assigns default theatre "PVR". 4. Full constructor → sets all details.

● Add method: printTicket() → displays ticket details.

● In main(): Create and print multiple tickets.

SOLUTION:

public class MovieTicket {

String movieName;

String theatreName;

int seatNumber;

double price;

// Default constructor

public MovieTicket() {

this("Unknown", "Unknown", -1, 0.0);

}

// Constructor with movie name

public MovieTicket(String movieName) {

this(movieName, "Unknown", -1, 200.0);

}

// Constructor with movie name and seat number

public MovieTicket(String movieName, int seatNumber) {

this(movieName, "PVR", seatNumber, 200.0);

}

// Full constructor

public MovieTicket(String movieName, String theatreName, int seatNumber, double price) {

this.movieName = movieName;

this.theatreName = theatreName;

this.seatNumber = seatNumber;

this.price = price;

}

public void printTicket() {

System.out.println("\n=== MOVIE TICKET ===");

System.out.println("Movie: " + movieName);

System.out.println("Theatre: " + theatreName);

System.out.println("Seat: " + seatNumber);

System.out.println("Price: Rs." + price);

}

public static void main(String[] args) {

MovieTicket t1 = new MovieTicket();

MovieTicket t2 = new MovieTicket("Inception");

MovieTicket t3 = new MovieTicket("Avengers", 12);

MovieTicket t4 = new MovieTicket("Interstellar", "IMAX", 25, 500.0);

t1.printTicket();

t2.printTicket();

t3.printTicket();

t4.printTicket();

}

}

OUTPUT:

=== MOVIE TICKET ===

Movie: Unknown

Theatre: Unknown

Seat: -1

Price: Rs.0.0

=== MOVIE TICKET ===

Movie: Inception

Theatre: Unknown

Seat: -1

Price: Rs.200.0

=== MOVIE TICKET ===

Movie: Avengers

Theatre: PVR

Seat: 12

Price: Rs.200.0

=== MOVIE TICKET ===

Movie: Interstellar

Theatre: IMAX

Seat: 25

Price: Rs.500.0

**Problem 2: Bank Account System** ��

Create a **Bank Account** management program.

● Class BankAccount with fields: String accountHolder, int accountNumber, double balance.

● Implement constructor overloading:

○ Default constructor → balance = 0.

○ Constructor with name → assigns random account number.

○ Constructor with name and initial balance → assigns both.

● Add methods:

1



○ deposit(double amount)

○ withdraw(double amount)

○ displayAccount()

● In main(): Create accounts, deposit/withdraw, and display balance.

SOLUTION:

import java.util.Random;

public class BankAccount {

String accountHolder;

int accountNumber;

double balance;

// Default constructor

public BankAccount() {

this("Unknown", 0.0);

}

// Constructor with name

public BankAccount(String accountHolder) {

this(accountHolder, 0.0);

}

// Constructor with name and initial balance

public BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.accountNumber = new Random().nextInt(900000) + 100000; // 6-digit account number

this.balance = balance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println(accountHolder + " deposited: Rs." + amount);

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println(accountHolder + " withdrew: Rs." + amount);

} else {

System.out.println("Insufficient balance for " + accountHolder);

}

}

public void displayAccount() {

System.out.println("\n=== ACCOUNT DETAILS ===");

System.out.println("Holder: " + accountHolder);

System.out.println("Account No: " + accountNumber);

System.out.println("Balance: Rs." + balance);

}

public static void main(String[] args) {

BankAccount a1 = new BankAccount();

BankAccount a2 = new BankAccount("Alice");

BankAccount a3 = new BankAccount("Bob", 5000.0);

a1.deposit(1000);

a2.deposit(2000);

a3.withdraw(1500);

a1.displayAccount();

a2.displayAccount();

a3.displayAccount();

}

}

OUTPUT:

Unknown deposited: Rs.1000.0

Alice deposited: Rs.2000.0

Bob withdrew: Rs.1500.0

=== ACCOUNT DETAILS ===

Holder: Unknown

Account No: 552401

Balance: Rs.1000.0

=== ACCOUNT DETAILS ===

Holder: Alice

Account No: 554796

Balance: Rs.2000.0

=== ACCOUNT DETAILS ===

Holder: Bob

Account No: 838813

Balance: Rs.3500.0

**Problem 3: Library Book Management** ��

Design a system for managing **Library Books**.

● Class Book with fields: String title, String author, String isbn, boolean isAvailable.

● Constructor overloading:

○ Default constructor → empty book.

○ Constructor with title and author.

○ Constructor with all details.

● Methods:

○ borrowBook() → sets available = false.

○ returnBook() → sets available = true.

○ displayBookInfo().

● In main(): Create books, borrow/return them, display info.

SOLUTION:

public class Book {

String title;

String author;

String isbn;

boolean isAvailable;

// Default constructor

public Book() {

this("Unknown", "Unknown", "N/A", true);

}

// Constructor with title and author

public Book(String title, String author) {

this(title, author, "N/A", true);

}

// Constructor with all details

public Book(String title, String author, String isbn, boolean isAvailable) {

this.title = title;

this.author = author;

this.isbn = isbn;

this.isAvailable = isAvailable;

}

public void borrowBook() {

if (isAvailable) {

isAvailable = false;

System.out.println(title + " has been borrowed.");

} else {

System.out.println(title + " is not available.");

}

}

public void returnBook() {

if (!isAvailable) {

isAvailable = true;

System.out.println(title + " has been returned.");

}

}

public void displayBookInfo() {

System.out.println("\n=== BOOK INFO ===");

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

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

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

System.out.println("Available: " + (isAvailable ? "Yes" : "No"));

}

public static void main(String[] args) {

Book b1 = new Book();

Book b2 = new Book("1984", "George Orwell");

Book b3 = new Book("The Hobbit", "J.R.R. Tolkien", "978-0547928227", true);

b1.displayBookInfo();

b2.displayBookInfo();

b3.displayBookInfo();

b2.borrowBook();

b2.displayBookInfo();

b2.returnBook();

b2.displayBookInfo();

}

}

OUTPUT:

=== BOOK INFO ===

Title: Unknown

Author: Unknown

ISBN: N/A

Available: Yes

=== BOOK INFO ===

Title: 1984

Author: George Orwell

ISBN: N/A

Available: Yes

=== BOOK INFO ===

Title: The Hobbit

Author: J.R.R. Tolkien

ISBN: 978-0547928227

Available: Yes

1984 has been borrowed.

=== BOOK INFO ===

Title: 1984

Author: George Orwell

ISBN: N/A

Available: No

1984 has been returned.

=== BOOK INFO ===

Title: 1984

Author: George Orwell

ISBN: N/A

Available: Yes

**Problem 4: Food Delivery Order** ��

Create a program to simulate a **Food Delivery System**.

2



● Class FoodOrder with fields: String customerName, String foodItem, int quantity, double price.

● Constructor overloading:

1. Default constructor → assigns "Unknown" order.

2. Constructor with food item → sets quantity = 1, price = default.

3. Constructor with food item and quantity → calculates price = quantity × fixedRate.

● Method: printBill() → displays order details and total price.

● In main(): Create multiple orders and print bills.

SOLUTION:

public class FoodOrder {

String customerName;

String foodItem;

int quantity;

double price;

static final double FIXED\_RATE = 150.0;

// Default constructor

public FoodOrder() {

this("Unknown", "Unknown", 0, 0.0);

}

// Constructor with food item

public FoodOrder(String foodItem) {

this("Unknown", foodItem, 1, FIXED\_RATE);

}

// Constructor with food item and quantity

public FoodOrder(String foodItem, int quantity) {

this("Unknown", foodItem, quantity, quantity \* FIXED\_RATE);

}

// Full constructor

public FoodOrder(String customerName, String foodItem, int quantity, double price) {

this.customerName = customerName;

this.foodItem = foodItem;

this.quantity = quantity;

this.price = price;

}

public void printBill() {

System.out.println("\n=== FOOD ORDER BILL ===");

System.out.println("Customer: " + customerName);

System.out.println("Food Item: " + foodItem);

System.out.println("Quantity: " + quantity);

System.out.println("Total Price: Rs." + price);

}

public static void main(String[] args) {

FoodOrder o1 = new FoodOrder();

FoodOrder o2 = new FoodOrder("Burger");

FoodOrder o3 = new FoodOrder("Pizza", 3);

FoodOrder o4 = new FoodOrder("Alice", "Pasta", 2, 400.0);

o1.printBill();

o2.printBill();

o3.printBill();

o4.printBill();

}

}

OUTPUT:

=== FOOD ORDER BILL ===

Customer: Unknown

Food Item: Unknown

Quantity: 0

Total Price: Rs.0.0

=== FOOD ORDER BILL ===

Customer: Unknown

Food Item: Burger

Quantity: 1

Total Price: Rs.150.0

=== FOOD ORDER BILL ===

Customer: Unknown

Food Item: Pizza

Quantity: 3

Total Price: Rs.450.0

=== FOOD ORDER BILL ===

Customer: Alice

Food Item: Pasta

Quantity: 2

Total Price: Rs.400.0

**Problem 5: Fitness Tracker** ��

Design a **Fitness Tracker App** simulation.

● Class Workout with fields: String activityName, int durationInMinutes, int caloriesBurned.

● Constructor overloading:

1. Default constructor → "Walking", 30 mins, 100 calories.

2. Constructor with activity name → assigns default duration.

3. Constructor with activity and duration → calculate caloriesBurned = duration × 5. ● Method: displayWorkout() → prints activity details.

● In main(): Create different workouts and display details.

3