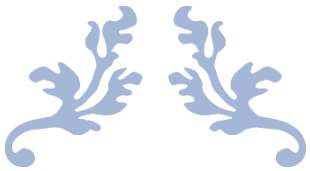


JAVA PROGRAMMING

1) Smart City Parking Management System

NOTE:

This program simulates a Smart City Parking Management System using HashMap to manage multiple parking lots. Each lot has a unique ID, total slots, and currently occupied slots. The system allows users to add parking lots, park/remove vehicles, check available slots, find the lot with most available spaces, and display all lots sorted by available spaces. All operations handle over-parking and empty lot removal gracefully with appropriate error messages.

AIM:

To design a comprehensive parking management system using Map interface with dynamic user input and proper error handling for real-world parking scenarios.

ALGORITHM:

1. Start the program.
2. Create ParkingLot class with lotId, totalSlots, and occupiedSlots attributes.
3. Use HashMap to store parking lots with lotId as key.
4. Display menu with 7 options.
5. Accept user choice and input dynamically.
6. For Add Parking Lot: Check if lot exists, if not add new ParkingLot object.
7. For Park Vehicle: Check if lot exists, if occupied < total, increment occupied.
8. For Remove Vehicle: Check if lot exists, if occupied > 0, decrement occupied.
9. For Display Available Slots: Show total, occupied, and available slots for specific lot.
10. For Most Available: Iterate through all lots using enhanced for loop to find maximum available slots.
11. For Display All Sorted: Copy values to ArrayList, sort by available slots in descending order using comparator, display all.
12. Handle invalid lot IDs with error messages.
13. Print registration number and close Scanner.
14. End the program.

CODE:

```
import java.util.*;

class ParkingLot {
    private String lotId;
    private int totalSlots;
    private int occupiedSlots;
```

```
    public ParkingLot(String lotId, int totalSlots) {
        this.lotId = lotId;
        this.totalSlots = totalSlots;
        this.occupiedSlots = 0;
    }

    public String getLotId() { return lotId; }
    public int getTotalSlots() { return totalSlots; }
    public int getOccupiedSlots() { return occupiedSlots; }
    public int getAvailableSlots() { return totalSlots - occupiedSlots; }

    public boolean parkVehicle() {
        if (occupiedSlots < totalSlots) {
            occupiedSlots++;
            return true;
        }
        return false;
    }

    public boolean removeVehicle() {
        if (occupiedSlots > 0) {
            occupiedSlots--;
            return true;
        }
        return false;
    }
}

public class ParkingManagementSystem {
    private static Map<String, ParkingLot> parkingLots = new HashMap<>();

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n========================================")
;
            System.out.println("  SMART CITY PARKING MANAGEMENT SYSTEM");
            System.out.println("========================================");
            System.out.println("1. Add Parking Lot");
            System.out.println("2. Park Vehicle");
            System.out.println("3. Remove Vehicle");
            System.out.println("4. Display Available Slots");
            System.out.println("5. Find Lot with Most Available Slots");
            System.out.println("6. Display All Lots Sorted by Available
Spaces");
```

```
            System.out.println("7. Exit");
            System.out.print("----------------------------------------------------------------------
\nEnter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1 -> addParkingLot(sc);
                case 2 -> parkVehicle(sc);
                case 3 -> removeVehicle(sc);
                case 4 -> displayAvailableSlots(sc);
                case 5 -> findMostAvailableLot();
                case 6 -> displayAllLotsSorted();
                case 7 -> System.out.println("Exiting...");
                default -> System.out.println("Invalid choice. Try again.");
            }

        } while (choice != 7);

        System.out.println("Reg. No: 23BBS0071");
        sc.close();
    }

    private static void addParkingLot(Scanner sc) {
        System.out.print("Enter Lot ID: ");
        String lotId = sc.nextLine();
        System.out.print("Enter Total Slots: ");
        int totalSlots = sc.nextInt();
        sc.nextLine();

        if (parkingLots.containsKey(lotId)) {
            System.out.println("Lot ID already exists!");
        } else {
            parkingLots.put(lotId, new ParkingLot(lotId, totalSlots));
            System.out.println("Parking lot added successfully!");
        }
    }

    private static void parkVehicle(Scanner sc) {
        System.out.print("Enter Lot ID: ");
        String lotId = sc.nextLine();

        if (!parkingLots.containsKey(lotId)) {
            System.out.println("Invalid Lot ID!");
        } else {
            if (parkingLots.get(lotId).parkVehicle()) {
                System.out.println("Vehicle parked successfully!");
            } else {
```

```
                System.out.println("Lot is full! Cannot park vehicle.");
            }
        }
    }

    private static void removeVehicle(Scanner sc) {
        System.out.print("Enter Lot ID: ");
        String lotId = sc.nextLine();

        if (!parkingLots.containsKey(lotId)) {
            System.out.println("Invalid Lot ID!");
        } else {
            if (parkingLots.get(lotId).removeVehicle()) {
                System.out.println("Vehicle removed successfully!");
            } else {
                System.out.println("Lot is empty! No vehicles to remove.");
            }
        }
    }

    private static void displayAvailableSlots(Scanner sc) {
        System.out.print("Enter Lot ID: ");
        String lotId = sc.nextLine();

        if (!parkingLots.containsKey(lotId)) {
            System.out.println("Invalid Lot ID!");
        } else {
            ParkingLot lot = parkingLots.get(lotId);
            System.out.println("\nLot ID: " + lot.getLotId());
            System.out.println("Total Slots: " + lot.getTotalSlots());
            System.out.println("Occupied Slots: " + lot.getOccupiedSlots());
            System.out.println("Available Slots: " + lot.getAvailableSlots());
        }
    }

    private static void findMostAvailableLot() {
        if (parkingLots.isEmpty()) {
            System.out.println("No parking lots available!");
            return;
        }

        ParkingLot maxLot = null;
        int maxSlots = -1;
        for (ParkingLot lot : parkingLots.values()) {
            if (lot.getAvailableSlots() > maxSlots) {
                maxSlots = lot.getAvailableSlots();
                maxLot = lot;
            }
```

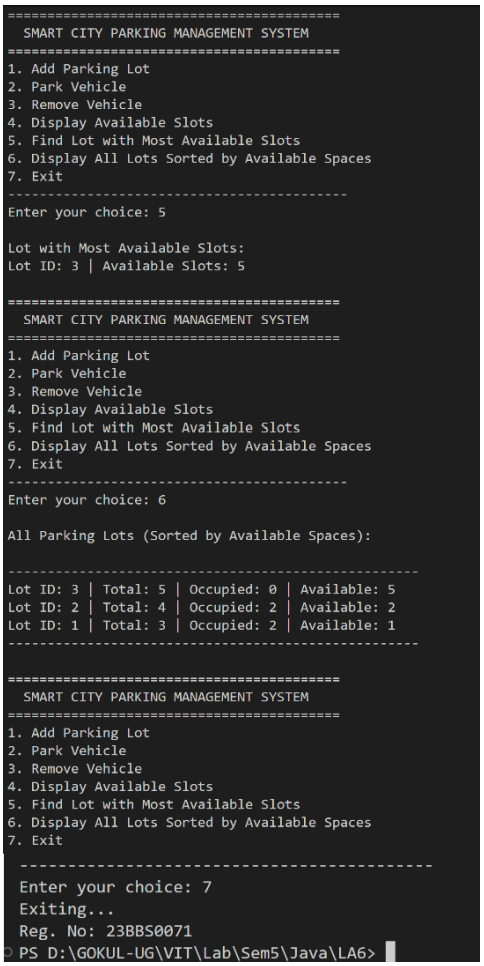
```
        }

        if (maxLot != null) {
            System.out.println("\nLot with Most Available Slots:");
            System.out.println("Lot ID: " + maxLot.getLotId() + " | Available
Slots: " + maxLot.getAvailableSlots());
        }
    }

    private static void displayAllLotsSorted() {
        if (parkingLots.isEmpty()) {
            System.out.println("No parking lots available!");
            return;
        }

        List<ParkingLot> sortedLots = new ArrayList<>(parkingLots.values());
        sortedLots.sort((a, b) -> Integer.compare(b.getAvailableSlots(),
a.getAvailableSlots()));

        System.out.println("\nAll Parking Lots (Sorted by Available
Spaces):\n");
        System.out.println("----------------------------------------------------------------------------
--");
        for (ParkingLot lot : sortedLots) {
            System.out.println("Lot ID: " + lot.getLotId() + " | Total: " +
lot.getTotalSlots() +
                    " | Occupied: " + lot.getOccupiedSlots() + " | Available:
" + lot.getAvailableSlots());
        }
        System.out.println("----------------------------------------------------------------------------
--");
    }
}
```

OUTPUT:

```
========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 1
Enter Lot ID: 1
Enter Total Slots: 3
Parking lot added successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 1
Enter Lot ID: 2
Enter Total Slots: 4
Parking lot added successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 1
Enter Lot ID: 3
Enter Total Slots: 5
Parking lot added successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 2
Enter Lot ID: 1
Vehicle parked successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 2
Enter Lot ID: 1
Vehicle parked successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
```

```
Enter your choice: 2
Enter Lot ID: 1
Vehicle parked successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 2
Enter Lot ID: 1
Lot is full! Cannot park vehicle.

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 2
Enter Lot ID: 2
Vehicle parked successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
```

```
Enter your choice: 2
Enter Lot ID: 2
Vehicle parked successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 3
Enter Lot ID: 1
Vehicle removed successfully!

========================================
  SMART CITY PARKING MANAGEMENT SYSTEM
========================================
1. Add Parking Lot
2. Park Vehicle
3. Remove Vehicle
4. Display Available Slots
5. Find Lot with Most Available Slots
6. Display All Lots Sorted by Available Spaces
7. Exit
----------------------------------------
Enter your choice: 4
Enter Lot ID: 1

Lot ID: 1
Total Slots: 3
Occupied Slots: 2
Available Slots: 1
```



2) Online Course Enrollment System

NOTE:

This program implements an Online Course Enrollment System using Set interface to manage student enrollments in multiple courses. Each course maintains a unique set of students with no duplicates. The system allows users to create courses, enroll/remove students, display enrolled students, find common students between courses, and find unique students. All operations use enhanced for loops and handle invalid course IDs gracefully.

AIM:

To design a course management system using Set interface with dynamic user input to manage unique student enrollments across multiple courses.

ALGORITHM:

1. Start the program.
2. Create a Map with course names as keys and HashSet<String> of student IDs as values.
3. Display menu with 7 options.

4. Accept user choice and input dynamically.
5. For Create Course: Check if course exists, if not add new HashSet for the course.
6. For Enroll Student: Check if course exists, attempt to add student using Set.add() method.
7. For Remove Student: Check if course exists, attempt to remove student using Set.remove() method.
8. For Display Students: Check if course exists, use enhanced for loop to iterate through student set and display.
9. For Common Students: Get two course names, create copy of first set, use retainAll() to find intersection, display using enhanced for loop.
10. For Unique Students: Create copies of both sets, use removeAll() on each to find differences, display both using enhanced for loops.
11. Handle invalid course IDs with error messages.
12. Print registration number and close Scanner.
13. End the program.

CODE:

```
import java.util.*;

public class CourseEnrollmentSystem {
    private static Map<String, Set<String>> courses = new HashMap<>();

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n========================================")
;
            System.out.println("  ONLINE COURSE ENROLLMENT SYSTEM");
            System.out.println("========================================");
            System.out.println("1. Create Course");
            System.out.println("2. Enroll Student");
            System.out.println("3. Remove Student");
            System.out.println("4. Display Students in Course");
            System.out.println("5. Find Common Students");
            System.out.println("6. Find Unique Students");
            System.out.println("7. Exit");
            System.out.print("----------------------------------------------------------------------
\nEnter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();
```

```
            switch (choice) {
                case 1 -> createCourse(sc);
                case 2 -> enrollStudent(sc);
                case 3 -> removeStudent(sc);
                case 4 -> displayCourseStudents(sc);
                case 5 -> findCommonStudents(sc);
                case 6 -> findUniqueStudents(sc);
                case 7 -> System.out.println("Exiting...");
                default -> System.out.println("Invalid choice. Try again.");
            }

        } while (choice != 7);

        System.out.println("Reg. No: 23BBS0071");
        sc.close();
    }

    private static void createCourse(Scanner sc) {
        System.out.print("Enter Course Name: ");
        String courseName = sc.nextLine();

        if (courses.containsKey(courseName)) {
            System.out.println("Course already exists!");
        } else {
            courses.put(courseName, new HashSet<>());
            System.out.println("Course created successfully!");
        }
    }

    private static void enrollStudent(Scanner sc) {
        System.out.print("Enter Course Name: ");
        String courseName = sc.nextLine();

        if (!courses.containsKey(courseName)) {
            System.out.println("Course does not exist!");
        } else {
            System.out.print("Enter Student ID: ");
            String studentId = sc.nextLine();

            if (courses.get(courseName).add(studentId)) {
                System.out.println("Student enrolled successfully!");
            } else {
                System.out.println("Student is already enrolled in this
course!");
            }
        }
    }
```

```
    private static void removeStudent(Scanner sc) {
        System.out.print("Enter Course Name: ");
        String courseName = sc.nextLine();

        if (!courses.containsKey(courseName)) {
            System.out.println("Course does not exist!");
        } else {
            System.out.print("Enter Student ID: ");
            String studentId = sc.nextLine();

            if (courses.get(courseName).remove(studentId)) {
                System.out.println("Student removed successfully!");
            } else {
                System.out.println("Student not found in this course!");
            }
        }
    }

    private static void displayCourseStudents(Scanner sc) {
        System.out.print("Enter Course Name: ");
        String courseName = sc.nextLine();

        if (!courses.containsKey(courseName)) {
            System.out.println("Course does not exist!");
        } else {
            Set<String> students = courses.get(courseName);
            if (students.isEmpty()) {
                System.out.println("No students enrolled in this course.");
            } else {
                System.out.println("\nStudents in " + courseName + ":");
                System.out.println("----------------------------------------------------------------
----------");
                for (String student : students) {
                    System.out.println("- " + student);
                }
                System.out.println("----------------------------------------------------------------
----------");
            }
        }
    }

    private static void findCommonStudents(Scanner sc) {
        System.out.print("Enter First Course Name: ");
        String course1 = sc.nextLine();
        System.out.print("Enter Second Course Name: ");
        String course2 = sc.nextLine();
```

```
        if (!courses.containsKey(course1) || !courses.containsKey(course2)) {
            System.out.println("One or both courses do not exist!");
        } else {
            Set<String> common = new HashSet<>(courses.get(course1));
            common.retainAll(courses.get(course2));

            System.out.println("\nCommon Students in " + course1 + " and " +
course2 + ":");
            System.out.println("----------------------------------------------------------------------
------");
            if (common.isEmpty()) {
                System.out.println("No common students found.");
            } else {
                for (String student : common) {
                    System.out.println("- " + student);
                }
            }
            System.out.println("----------------------------------------------------------------------
------");
        }
    }

    private static void findUniqueStudents(Scanner sc) {
        System.out.print("Enter First Course Name: ");
        String course1 = sc.nextLine();
        System.out.print("Enter Second Course Name: ");
        String course2 = sc.nextLine();

        if (!courses.containsKey(course1) || !courses.containsKey(course2)) {
            System.out.println("One or both courses do not exist!");
        } else {
            Set<String> unique1 = new HashSet<>(courses.get(course1));
            unique1.removeAll(courses.get(course2));

            Set<String> unique2 = new HashSet<>(courses.get(course2));
            unique2.removeAll(courses.get(course1));

            System.out.println("\nUnique Students in " + course1 + ":");
            System.out.println("----------------------------------------------------------------------
------");
            if (unique1.isEmpty()) {
                System.out.println("No unique students.");
            } else {
                for (String student : unique1) {
                    System.out.println("- " + student);
                }
            }
```

```
            System.out.println("\nUnique Students in " + course2 + ":");
            System.out.println("
-------");
            if (unique2.isEmpty()) {
                System.out.println("No unique students.");
            } else {
                for (String student : unique2) {
                    System.out.println("- " + student);
                }
            }
            System.out.println("--------------------------------------------------------------------------------
-------");
        }
    }
}
```

OUTPUT:

```
========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 1
Enter Course Name: Java
Course created successfully!

========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 1
Enter Course Name: Python
Course created successfully!

========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 1
Enter Course Name: Deutsch
Course created successfully!
```

```
========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 2
Enter Course Name: Java
Enter Student ID: 001
Student enrolled successfully!

========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 2
Enter Course Name: Python
Enter Student ID: 123
Student enrolled successfully!

========================================
  ONLINE COURSE ENROLLMENT SYSTEM
========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
----------------------------------------
Enter your choice: 2
Enter Course Name: Deutsch
Enter Student ID: 071
Student enrolled successfully!
```

```
==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 3
Enter Course Name: Java
Enter Student ID: 001
Student removed successfully!

==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 4
Enter Course Name: Deutsch

Students in Deutsch:
--------------------------------------------------
- 071
--------------------------------------------------
```

```
==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 2
Enter Course Name: Deutsch
Enter Student ID: 123
Student enrolled successfully!

==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 5
Enter First Course Name: Deutsch
Enter Second Course Name: Python

Common Students in Deutsch and Python:
--------------------------------------------------
- 123
--------------------------------------------------

==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
Enter Second Course Name: Python

Common Students in Deutsch and Python:
--------------------------------------------------
- 123
--------------------------------------------------
```

```
==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================

Common Students in Deutsch and Python:
--------------------------------------------------
- 123
--------------------------------------------------

==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 6
Enter First Course Name: Deutsch
Enter Second Course Name: Python

Unique Students in Deutsch:
--------------------------------------------------
- 071

Unique Students in Python:
--------------------------------------------------
No unique students.
--------------------------------------------------
```

```
==========================================
  ONLINE COURSE ENROLLMENT SYSTEM
==========================================
1. Create Course
2. Enroll Student
3. Remove Student
4. Display Students in Course
5. Find Common Students
6. Find Unique Students
7. Exit
------------------------------------------
Enter your choice: 7
Exiting...
Reg. No: 23BBS0071
PS D:\GOKUL-UG\VIT\Lab\Sem5\Java\LA6>
```

3) Online Order Processing System

NOTE:

This program simulates an Online Order Processing System using Servlets. It accepts customer name, item name, quantity, and price from user input via HTML form. The OrderServlet validates inputs, calculates total amount, and forwards to BillServlet which generates an invoice in HTML format with formatted bill display. The system handles validation errors gracefully and displays appropriate error messages for invalid inputs.

AIM:

To design a web-based order processing system using Servlets with form validation, request forwarding, and formatted bill generation using dynamic user inputs.

ALGORITHM:

1. Start the program with HTML form in browser.
2. Create HTML form (order.html) with fields for customer name, item name, quantity, and price.
3. Accept user input through HTML form and submit to OrderServlet.
4. In OrderServlet, retrieve parameters using getParameter() method.
5. Validate all fields are not empty.
6. Check if quantity and price are positive numbers, throw NumberFormatException if invalid.
7. If validation fails, set error message in request attributes and include order.html.
8. If validation passes, calculate totalAmount = quantity × price.
9. Set all attributes (customerName, itemName, quantity, price, totalAmount) in request.
10. Use forward() to BillServlet.
11. In BillServlet, retrieve all attributes from request.
12. Get current timestamp using SimpleDateFormat.
13. Generate formatted HTML bill with CSS styling.
14. Display customer details, item details, quantity, unit price, and total amount.
15. Add button to place another order by linking to order.html.
16. End the program.

CODE:

src\main\java\BillServlet.java:

```
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class BillServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String customerName = (String) request.getAttribute("customerName");
        String itemName = (String) request.getAttribute("itemName");
        Integer quantity = (Integer) request.getAttribute("quantity");
        Double price = (Double) request.getAttribute("price");
        Double totalAmount = (Double) request.getAttribute("totalAmount");

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String timestamp = sdf.format(new Date());

        out.println("<html>");
        out.println("<head>");
        out.println("<title>Order Bill</title>");
        out.println("<style>");
        out.println("body { font-family: Arial, sans-serif; margin: 20px; }");
        out.println(".bill-container { border: 2px solid black; padding: 20px;
width: 400px; margin: 0 auto; }");
        out.println(".bill-header { text-align: center; font-weight: bold;
font-size: 18px; margin-bottom: 20px; }");
        out.println(".bill-content { margin-bottom: 10px; }");
        out.println(".bill-line { border-bottom: 1px solid #ccc; margin: 10px
0; }");
        out.println(".bill-total { font-weight: bold; font-size: 16px; text-
align: right; margin-top: 15px; }");
        out.println("</style>");
        out.println("</head>");
        out.println("<body>");

        out.println("<div class='bill-container'>");
        out.println("<div class='bill-header'>ORDER BILL</div>");
        out.println("<div class='bill-content'><strong>Bill Date &
Time:</strong> " + timestamp + "</div>");
        out.println("<div class='bill-content'><strong>Customer Name:</strong>
" + customerName + "</div>");
        out.println("<div class='bill-content'><strong>Item Name:</strong> " +
itemName + "</div>");
        out.println("<div  class='bill-line'></div>");
```

```
        out.println("<div class='bill-content'>Quantity: " + quantity +
"</div>");
        out.println("<div class='bill-content'>Unit Price: Rs. " +
String.format("%.2f", price) + "</div>");
        out.println("<div class='bill-line'></div>");
        out.println("<div class='bill-total'>Total Amount: Rs. " +
String.format("%.2f", totalAmount) + "</div>");
        out.println("<div style='text-align: center; margin-top: 20px;'>");
        out.println("<a href='order.html' style='text-decoration:
none;'><button>Place Another Order</button></a>");
        out.println("</div>");
        out.println("</div>");

        out.println("</body>");
        out.println("</html>");
    }
}
```

src\main\java\OrderServlet.java:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class OrderServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        String customerName = request.getParameter("customerName");
        String itemName = request.getParameter("itemName");
        String quantityStr = request.getParameter("quantity");
        String priceStr = request.getParameter("price");

        if (customerName == null || customerName.isEmpty() ||
            itemName == null || itemName.isEmpty() ||
            quantityStr == null || quantityStr.isEmpty() ||
            priceStr == null || priceStr.isEmpty()) {

            request.setAttribute("error", "All fields are required!");
            RequestDispatcher rd = request.getRequestDispatcher("order.html");
            rd.include(request, response);
            return;
        }

        try {
            int quantity = Integer.parseInt(quantityStr);
```

```
            double price = Double.parseDouble(priceStr);

            if (quantity <= 0 || price <= 0) {
                request.setAttribute("error", "Quantity and Price must be
positive!");
                RequestDispatcher rd =
request.getRequestDispatcher("order.html");
                rd.include(request, response);
                return;
            }

            double totalAmount = quantity * price;

            request.setAttribute("customerName", customerName);
            request.setAttribute("itemName", itemName);
            request.setAttribute("quantity", quantity);
            request.setAttribute("price", price);
            request.setAttribute("totalAmount", totalAmount);

            RequestDispatcher rd =
request.getRequestDispatcher("BillServlet");
            rd.forward(request, response);

        } catch (NumberFormatException e) {
            request.setAttribute("error", "Quantity must be integer and Price
must be decimal!");
            RequestDispatcher rd = request.getRequestDispatcher("order.html");
            rd.include(request, response);
        }
    }
}
```

src\main\webapp\WEB-INF\web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

    <servlet>
        <servlet-name>OrderServlet</servlet-name>
        <servlet-class>OrderServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>OrderServlet</servlet-name>
```

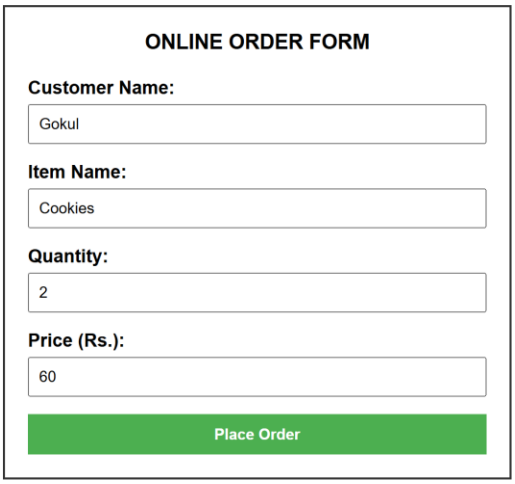
```
        <url-pattern>/OrderServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>BillServlet</servlet-name>
        <servlet-class>BillServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>BillServlet</servlet-name>
        <url-pattern>/BillServlet</url-pattern>
    </servlet-mapping>

</web-app>
```

src\main\webapp\order.jsp:

```
<!DOCTYPE html>
<html>
<head>
    <title>Order Processing System</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .form-container { border: 2px solid #333; padding: 20px; width: 400px;
margin: 0 auto; }
        .form-header { text-align: center; font-weight: bold; font-size: 18px;
margin-bottom: 20px; }
        .form-group { margin-bottom: 15px; }
        label { display: block; margin-bottom: 5px; font-weight: bold; }
        input[type="text"], input[type="number"] { width: 100%; padding: 8px;
box-sizing: border-box; }
        button { width: 100%; padding: 10px; background-color: #4CAF50; color:
white; font-weight: bold; border: none; cursor: pointer; }
        button:hover { background-color: #45a049; }
        .error { color: red; margin-bottom: 10px; font-weight: bold; }
    </style>
</head>
<body>
    <div class="form-container">
        <div class="form-header">ONLINE ORDER FORM</div>

        <%
            String error = (String) request.getAttribute("error");
            if (error != null) {
                out.println("<div class='error'>Error: " + error + "</div>");
            }
        %>
```

```
        <form action="OrderServlet" method="POST">
            <div class="form-group">
                <label for="customerName">Customer Name:</label>
                <input type="text" id="customerName" name="customerName"
required>
            </div>

            <div class="form-group">
                <label for="itemName">Item Name:</label>
                <input type="text" id="itemName" name="itemName" required>
            </div>

            <div class="form-group">
                <label for="quantity">Quantity:</label>
                <input type="number" id="quantity" name="quantity" required>
            </div>

            <div class="form-group">
                <label for="price">Price (Rs.):</label>
                <input type="number" id="price" name="price" step="0.01"
required>
            </div>

            <button type="submit">Place Order</button>
        </form>
    </div>
</body>
</html>
```

OUTPUT:



ORDER BILL

Bill Date & Time: 2025-10-31 15:33:43

Customer Name: Gokul

Item Name: Cookies

Quantity: 2

Unit Price: Rs. 60.00

Total Amount: Rs. 120.00

Place Another Order