**DIGITAL NURTURE 4.0 DEEP SKILLING JAVA FSE-WEEK1**

**NAME: SIVITHA GUNASEKARAN**

**SUPERSET ID: 6413354**

**WEEK 1: ALGORITHMS DATA STRUCTURES**

**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Steps:**

1. **Understand Sorting Algorithms:**
   * Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).
2. **Setup:**
   * Create a class **Order** with attributes like **orderId**, **customerName**, and **totalPrice**.
3. **Implementation:**
   * Implement **Bubble Sort** to sort orders by **totalPrice**.
   * Implement **Quick Sort** to sort orders by **totalPrice**.
4. **Analysis:**
   * Compare the performance (time complexity) of Bubble Sort and Quick Sort.
   * Discuss why Quick Sort is generally preferred over Bubble Sort.

**CODE SAMPLES:**

**package** demo;

**class** Order {

**int** orderId;

String customerName;

**double** totalPrice;

**public** Order(**int** orderId, String customerName, **double** totalPrice) {

**this**.orderId = orderId;

**this**.customerName = customerName;

**this**.totalPrice = totalPrice;

}

**public** String toString() {

**return** orderId + " | " + customerName + " | ₹" + totalPrice;

}

}

**public** **class** CustomerOrderSorting {

**public** **static** **void** bubbleSort(Order[] orders) {

**int** n = orders.length;

**for** (**int** i = 0; i < n - 1; i++) {

**for** (**int** j = 0; j < n - i - 1; j++) {

**if** (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

**public** **static** **void** quickSort(Order[] orders, **int** low, **int** high) {

**if** (low < high) {

**int** pi = *partition*(orders, low, high);

*quickSort*(orders, low, pi - 1);

*quickSort*(orders, pi + 1, high);

}

}

**public** **static** **int** partition(Order[] orders, **int** low, **int** high) {

**double** pivot = orders[high].totalPrice;

**int** i = low - 1;

**for** (**int** j = low; j < high; j++) {

**if** (orders[j].totalPrice < pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

orders[i + 1] = orders[high];

orders[high] = temp;

**return** i + 1;

}

**public** **static** **void** printOrders(Order[] orders, String title) {

System.***out***.println("\n" + title);

**for** (Order o : orders) {

System.***out***.println(o);

}

}

**public** **static** **void** main(String[] args) {

Order[] orders = {

**new** Order(101, "Ravi", 2500),

**new** Order(102, "Meena", 1200),

**new** Order(103, "Kiran", 4700),

**new** Order(104, "John", 1900)

};

Order[] bubbleSorted = orders.clone();

*bubbleSort*(bubbleSorted);

*printOrders*(bubbleSorted, "Bubble Sorted Orders:");

Order[] quickSorted = orders.clone();

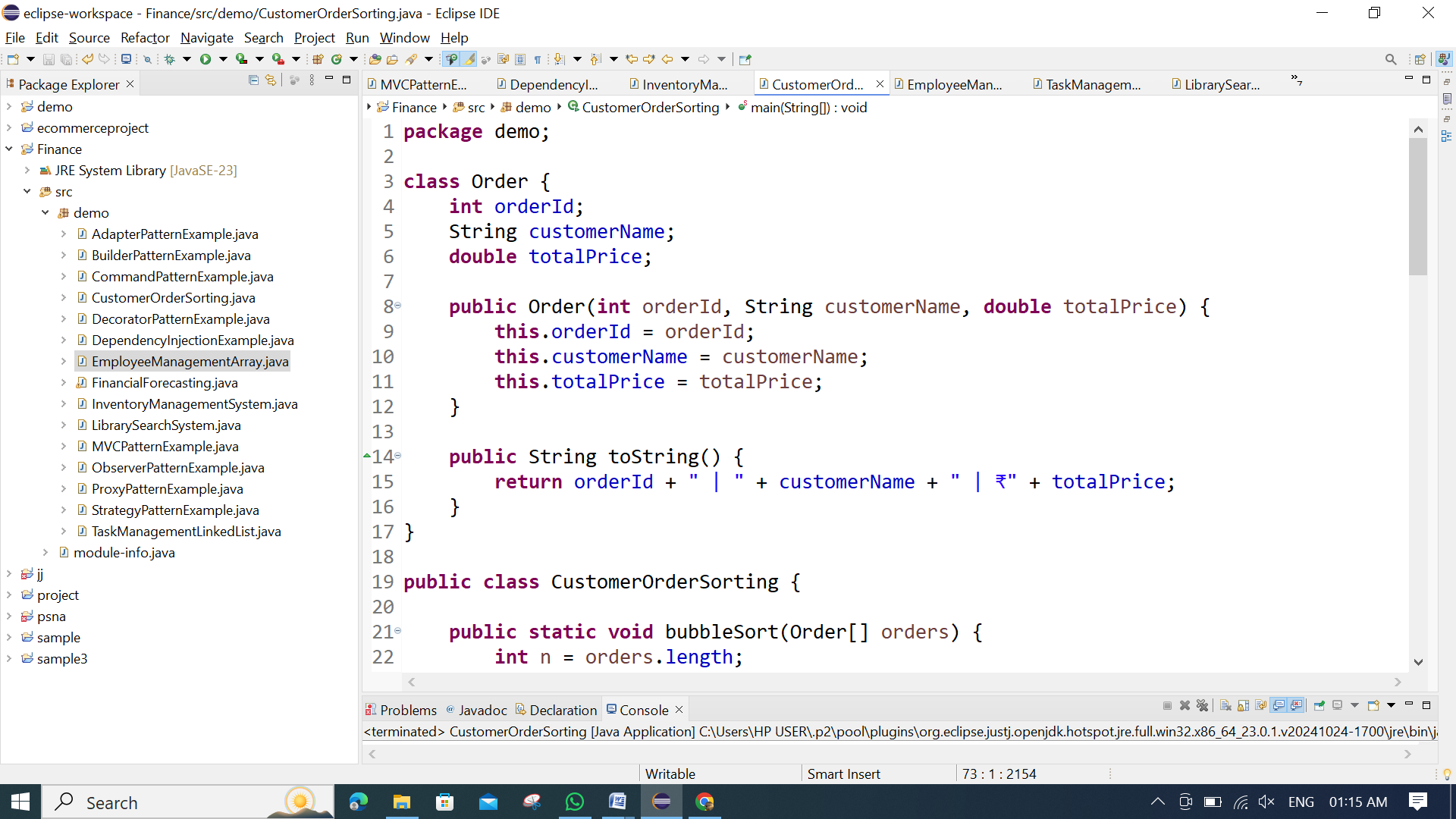
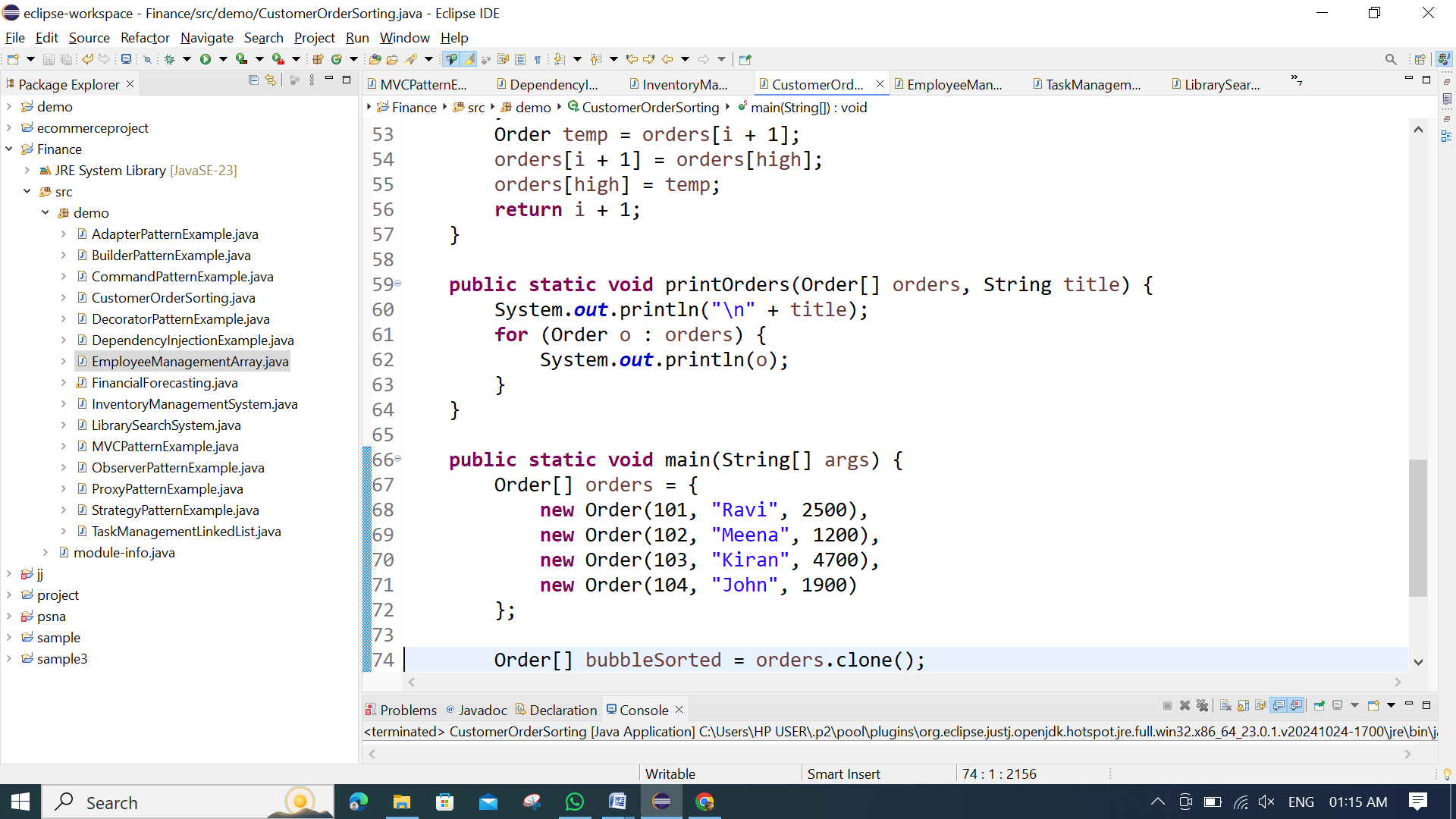
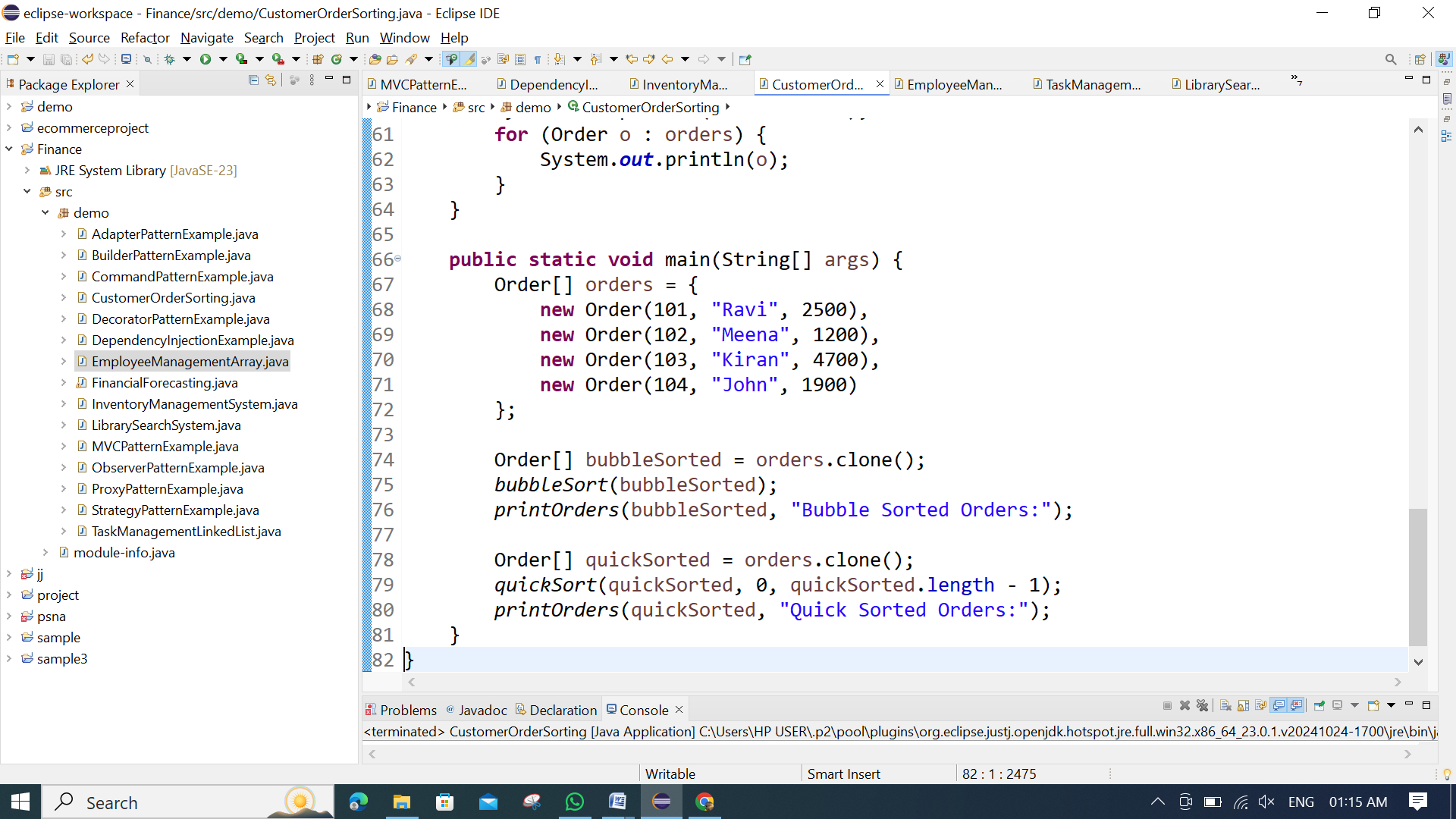
*quickSort*(quickSorted, 0, quickSorted.length - 1);

*printOrders*(quickSorted, "Quick Sorted Orders:");

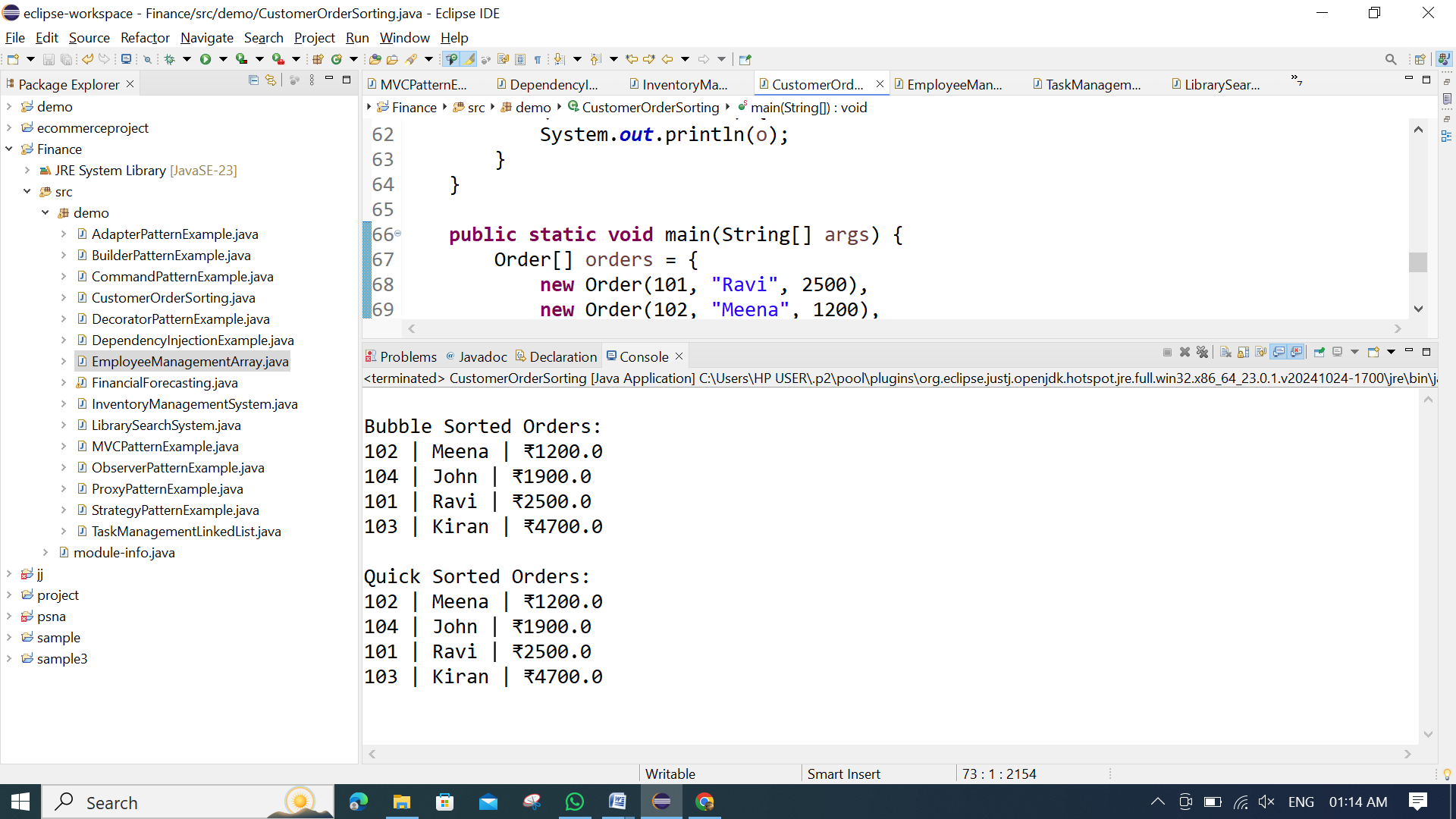
}

}

**MY SCREENSHOT PROOFS:**

**  **

**OUTPUT:**

****