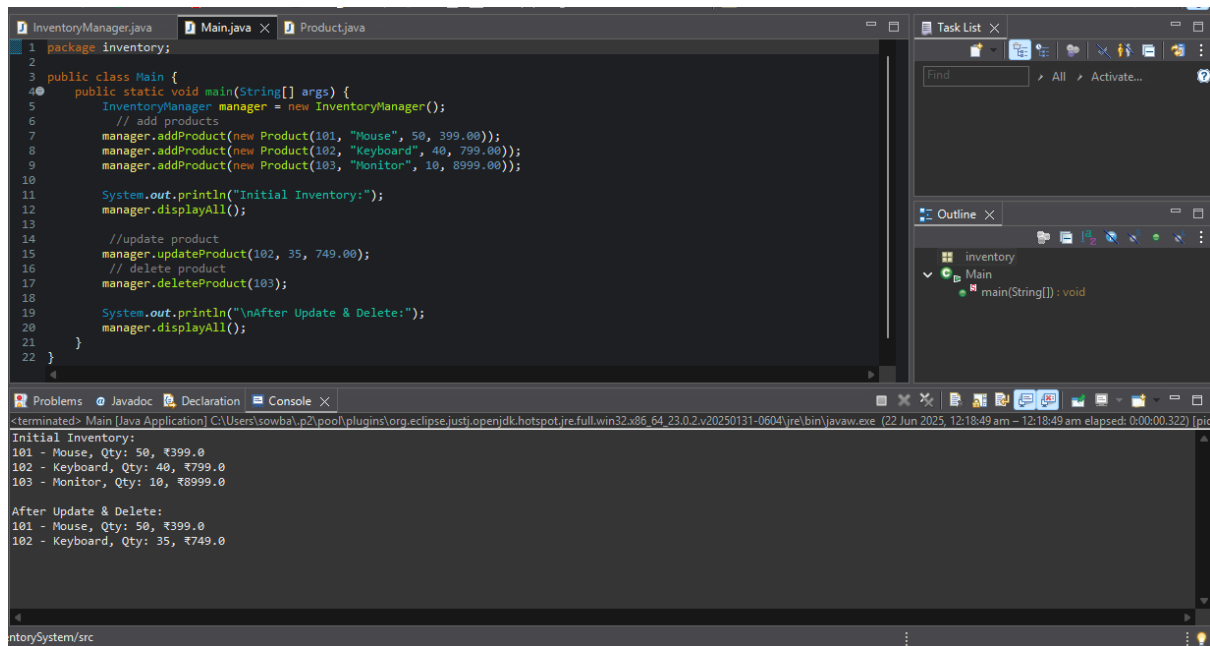


## Exercise 1: Inventory Management System

### OUTPUT:



The screenshot displays an IDE with three open files: `InventoryManager.java`, `Main.java`, and `Product.java`. The `Main.java` file contains the following code:

```
1 package inventory;
2
3 public class Main {
4     public static void main(String[] args) {
5         InventoryManager manager = new InventoryManager();
6         // add products
7         manager.addProduct(new Product(101, "Mouse", 50, 399.00));
8         manager.addProduct(new Product(102, "Keyboard", 40, 799.00));
9         manager.addProduct(new Product(103, "Monitor", 10, 8999.00));
10
11         System.out.println("Initial Inventory:");
12         manager.displayAll();
13
14         //update product
15         manager.updateProduct(102, 35, 749.00);
16         // delete product
17         manager.deleteProduct(103);
18
19         System.out.println("\nAfter Update & Delete:");
20         manager.displayAll();
21     }
22 }
```

The console output shows the execution results:

```
<terminated> Main [Java Application] C:\Users\sowba\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (22 Jun 2025, 12:18:49 am - 12:18:49 am elapsed: 0:00:00.322) [pid
Initial Inventory:
101 - Mouse, Qty: 50, ₹399.0
102 - Keyboard, Qty: 40, ₹799.0
103 - Monitor, Qty: 10, ₹8999.0

After Update & Delete:
101 - Mouse, Qty: 50, ₹399.0
102 - Keyboard, Qty: 35, ₹749.0
```

In the Inventory Management System, using efficient data structures like HashMap ensures fast add, update, and delete operations. This improves performance when handling large inventories. Choosing the right structure and analyzing time complexity helps build a system that is both scalable and optimized.