CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Project Based Learning in Java

Experiment 6

23CSP-304

Submitted To:
Faculty Name: Er. Deep Prakash

Submitted By:
Name: Garvi Dabas
UID: 23BCS11346
Section: KRG - 2B
Semester: 5th

Experiment 6: Product Data Processing using Java Streams

Aim

To process a large dataset of products using Java Stream API and perform operations such as grouping products by category, finding the most expensive product in each category, and calculating the average price of all products.

Objectives

- Implement a Product class with attributes: id, name, price, and category.
- Use **Stream API** for efficient data processing.
- Apply Collectors.groupingBy() to group products by category.
- Use Collectors.maxBy() to find the most expensive product per category.
- Use Collectors.averagingDouble() to calculate the average price of all products.
- Demonstrate advanced stream operations in a functional programming style.

Code Implementation

```
import java.util.*;
import java.util.stream.*;

class Product {
private int id;
private String name;
private double price;
private String category;

// Constructor
public Product(int id, String name, double price, String category) {
this.id = id;
this.name = name;
this.price = price;
this.category = category;
}

// Getters
public int getId() { return id; }
public String getName() { return name; }
public double getPrice() { return price; }
public String getCategory() { return category; }

@Override
public String toString() {
return name + " (₹" + price + ")";
}
}
public class StreamProductAnalysis {
public static void main(String[] args) {
```

```
List<Product> products = Arrays.asList(
new Product(1, "Laptop", 80000, "Electronics"),
new Product(2, "Smartphone", 25000, "Electronics"),
new Product(3, "Tablet", 30000, "Electronics"),
new Product(4, "Office Chair", 12000, "Furniture"),
new Product(5, "Desk", 7000, "Furniture"),
new Product(6, "Bookshelf", 9000, "Furniture"),
new Product(7, "Refrigerator", 45000, "Appliances"),
new Product(8, "Microwave", 18000, "Appliances")
);

// Step 2: Group products by category and find the most expensive one
Map<String, Optional<Product>> mostExpensiveByCategory = products.stream()
.collect(Collectors.groupingBy(
Product::getCategory,
Collectors.maxBy(Comparator.comparingDouble(Product::getPrice))
));

// Step 3: Calculate average price of all products
double averagePrice = products.stream()
.collect(Collectors.averagingDouble(Product::getPrice));

// Step 4: Display the results
mostExpensiveByCategory.forEach((category, product) ->
System.out.println(category + " → Most Expensive: " + product.get().getName() +
" (₹" + product.get().getPrice() + ")")
);

System.out.println("Average Price of All Products: ₹" + averagePrice);
}
}
```

Output

```
Electronics → Most Expensive: Laptop (₹80000.0)
Furniture → Most Expensive: Office Chair (₹12000.0)
Appliances → Most Expensive: Refrigerator (₹45000.0)
Average Price of All Products: ₹28250.0
```

Learning Outcomes

- Learned how to **group, aggregate, and analyze** data using Stream API.
- Understood the use of groupingBy(), maxBy(), and averagingDouble().
- Gained practical experience using **lambda expressions** and **functional programming** in Java.