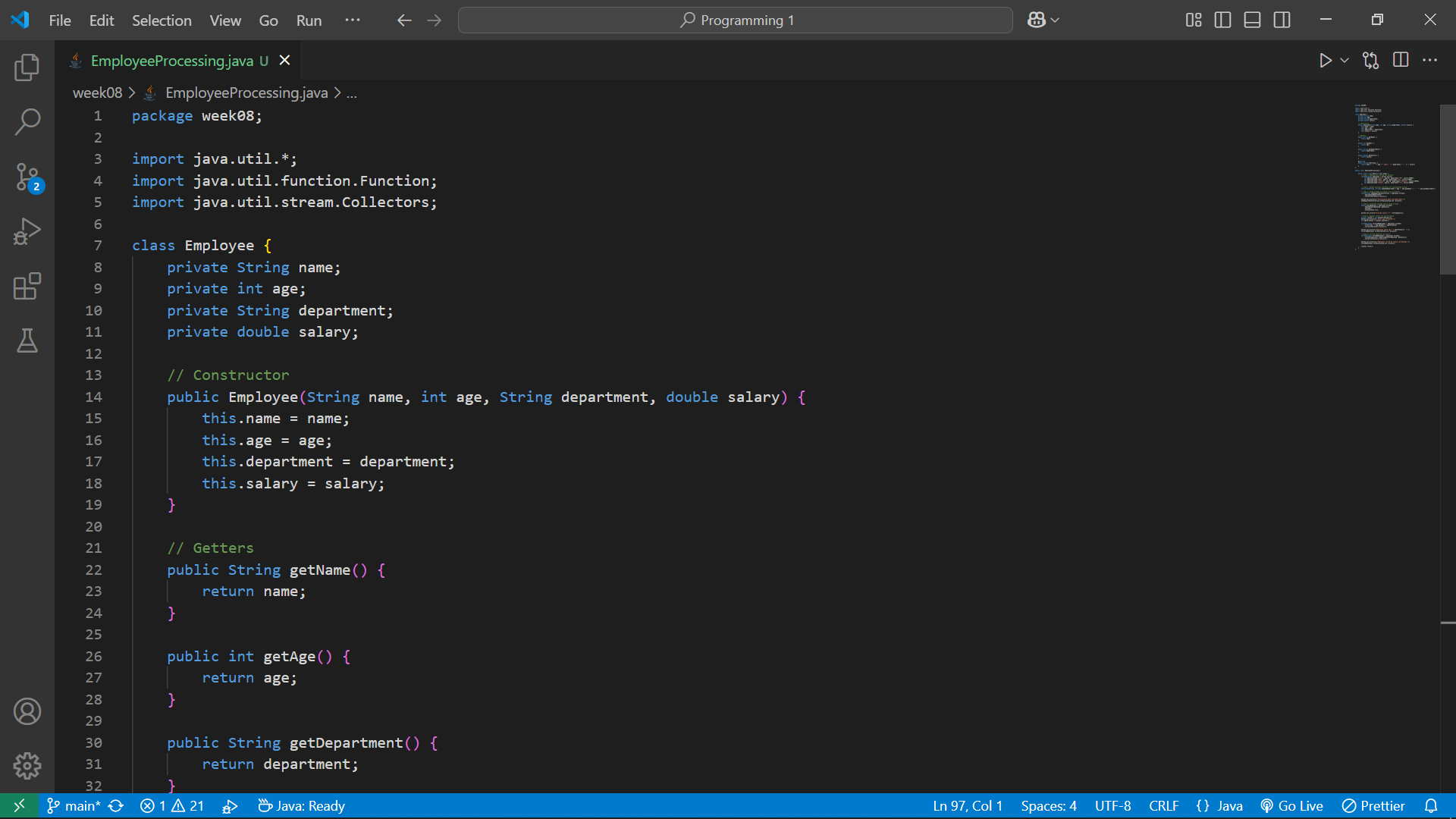
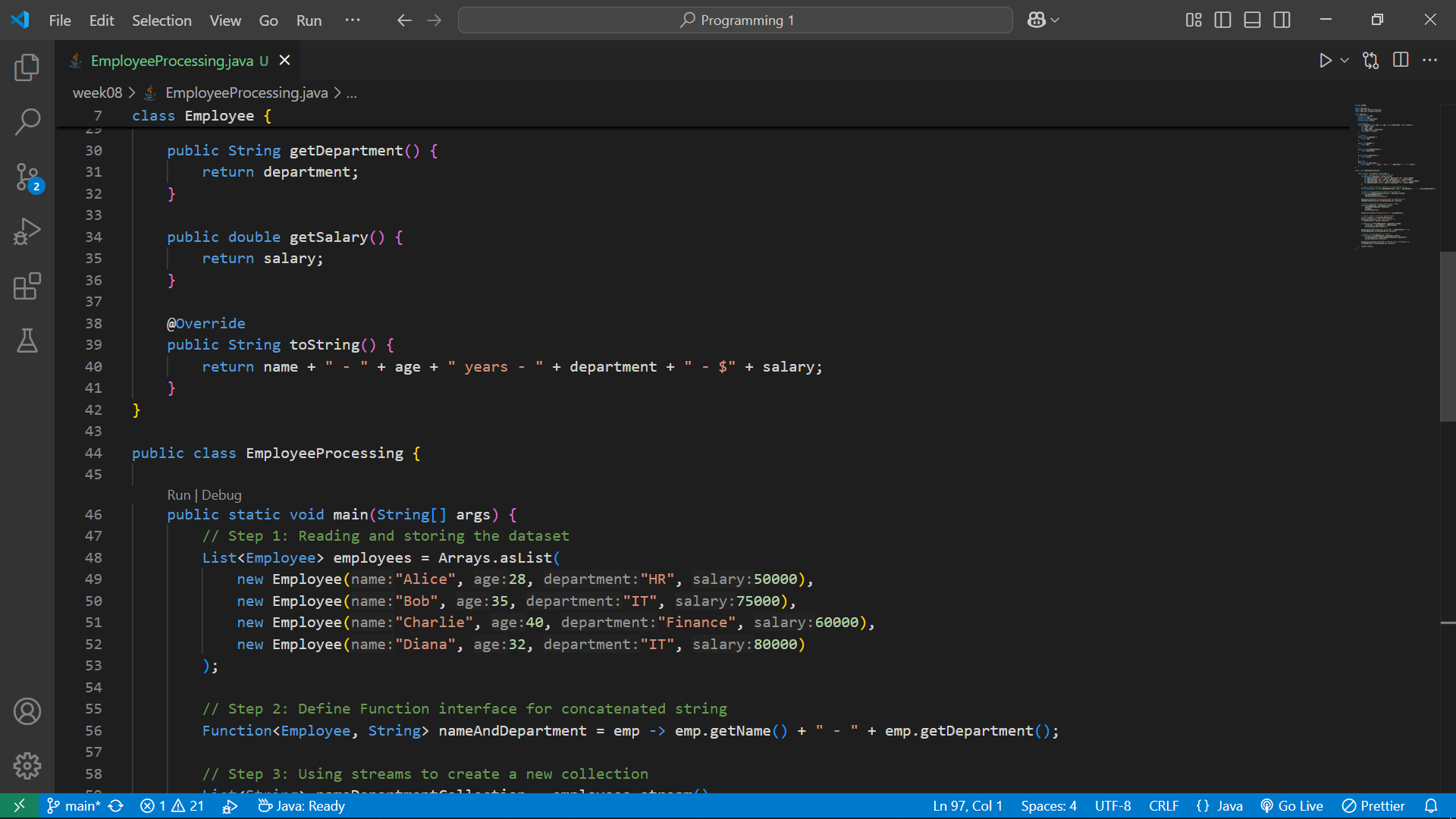
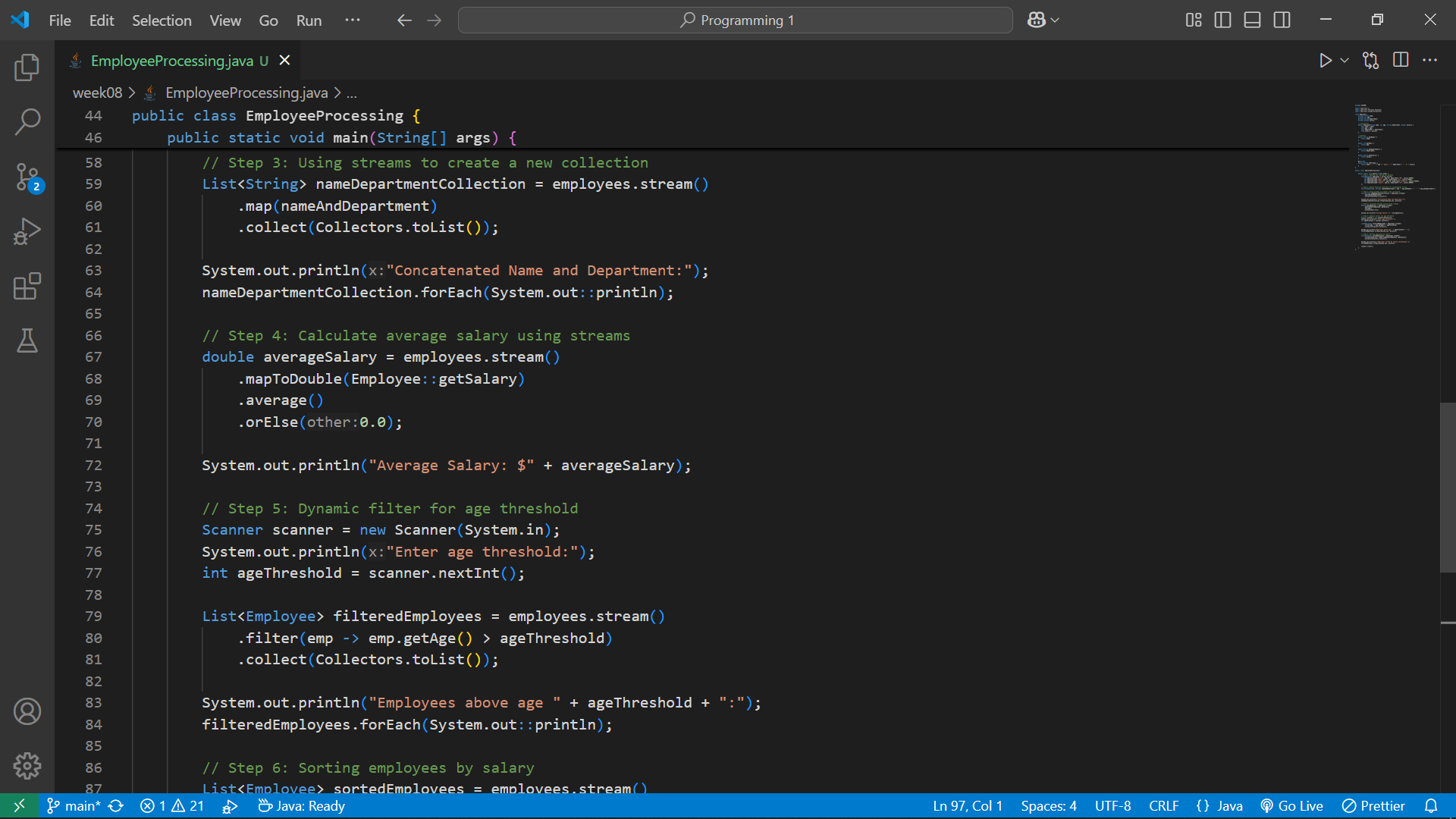
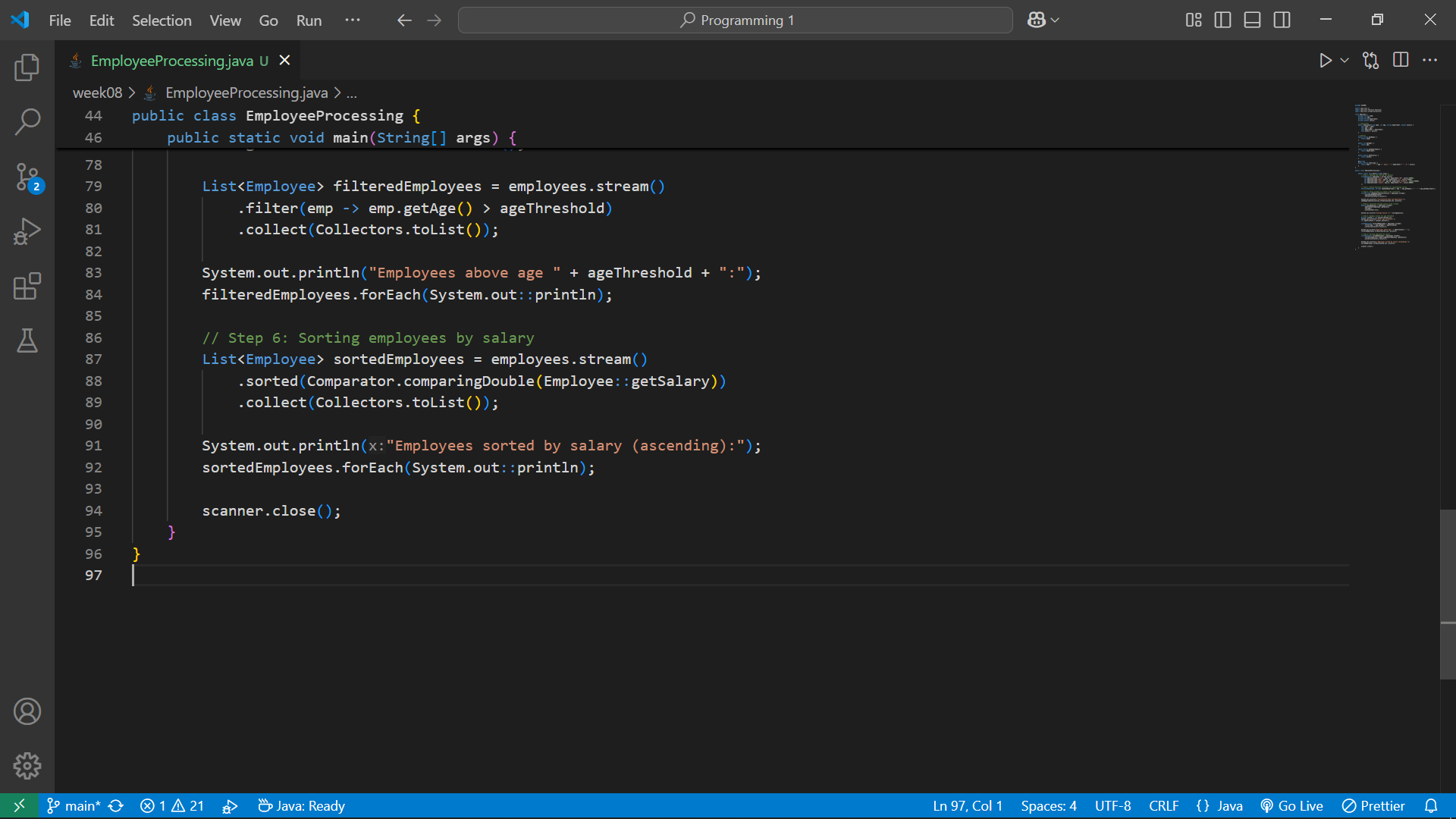
**Lewis Ochieng’ Ombaka - Programming Assignment Unit 8**









### **Explanation of Code Components**

#### **Program Structure and Correctness**

* **Dataset Storage:** The program uses a List<Employee> to store the dataset, ensuring it is accessible and manageable.
* **Concatenation Function:** Implements the Function interface to concatenate an employee's name and department, showcasing its utility.
* **Stream Operations:** Generates a new collection and calculates average salary effectively. The filter operation seamlessly incorporates the age threshold criterion.

#### **Proper Usage of Function Interface and Streams**

* The Function interface is clearly demonstrated through nameAndDepartment, transforming Employee objects into specific outputs.

### **Explanation of Additional Features**

#### **Dynamic Filters**

* **Implementation:** Instead of hardcoding the age threshold, the program now dynamically accepts it from user input using Scanner. This makes the filtering process more flexible and interactive.
* **Purpose:** Users can specify any age threshold, making the program adaptable to different scenarios without modifying the code.
* **Enhancement:** The feature enhances usability by empowering users to customize the filtering condition.

#### **Sorting**

* **Implementation:** The program uses the sorted() method with Comparator.comparingDouble(Employee::getSalary) to sort employees by salary in ascending order.
* **Purpose:** Sorting helps organize data for better visualization or analysis, such as identifying the highest or lowest salary earners.
* **Enhancement:** Sorting demonstrates proficiency with streams and adds a valuable analytical dimension to the program.
* Streams are employed for multiple operations: map() for transformation, filter() for conditional filtering, and mapToDouble() for numerical aggregation.

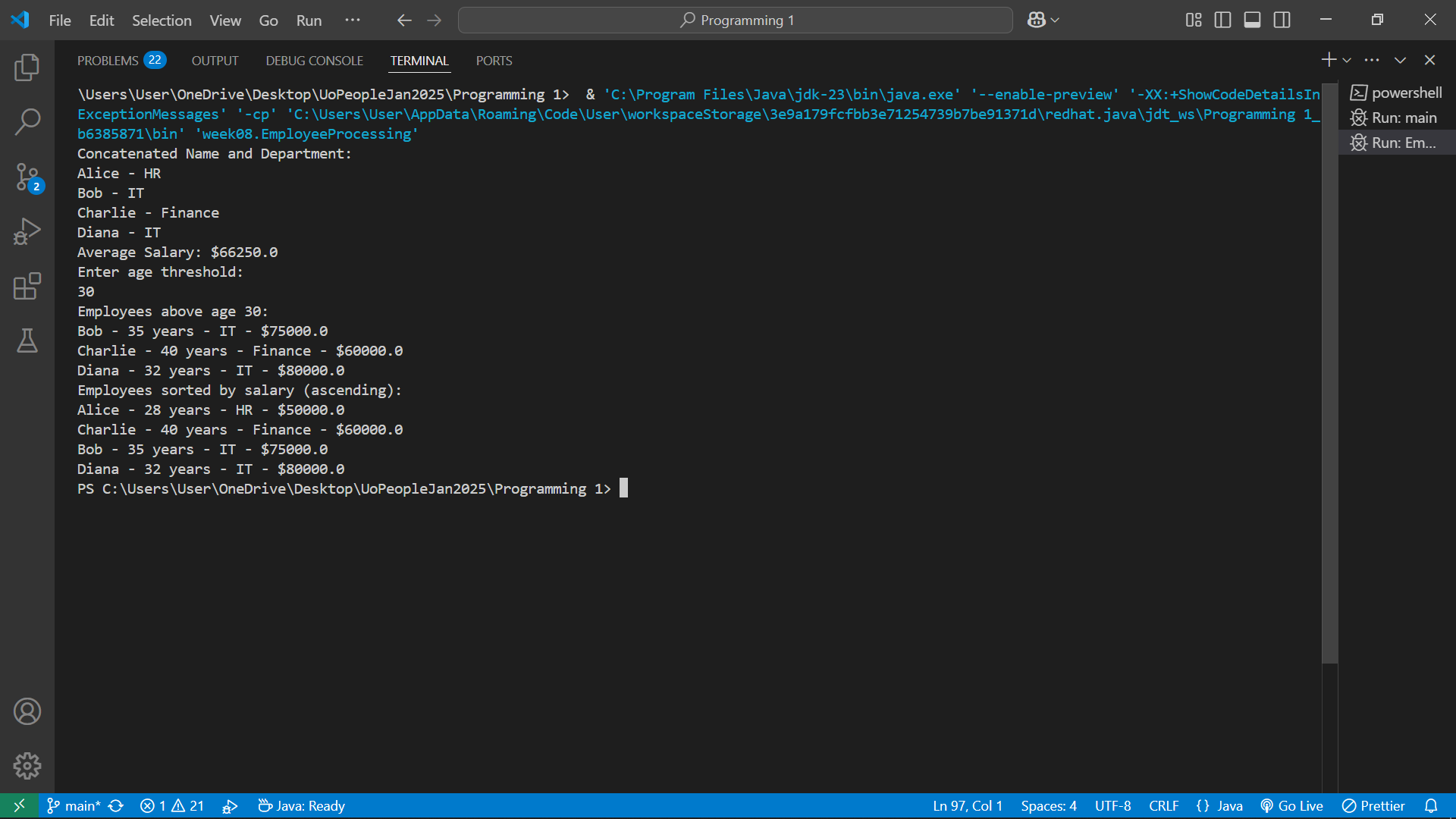
#### **Efficiency and Performance**

* The program takes advantage of stream operations, leveraging lazy evaluation for filter() and efficient processing for average().
* Processing is optimized by chaining stream operations, avoiding redundant iterations.

#### **Code Readability and Documentation**

* **Structure:** The code follows standard indentation practices and uses meaningful variable names.
* **Comments:** Each step is documented to explain the logic and purpose of complex operations.
* **Enhancements:** The overridden toString() method in the Employee class improves readability when printing objects.

### **Output**



### **Concatenated Name and Department**

This step creates a list where each employee's name is concatenated with their department using the Function interface. The resulting collection is printed:

Concatenated Name and Department:

Alice - HR

Bob - IT

Charlie - Finance

Diana - IT

* Each employee object from the dataset is processed, combining the name and department attributes (e.g., Alice from HR becomes "Alice - HR").
* This operation illustrates the Function interface in action, producing meaningful, streamlined data for display or further processing.

### **2. Average Salary**

The program calculates the average salary of all employees using a stream operation (mapToDouble() followed by average()). The result is displayed:

Average Salary: $66250.0

* This is the mean value of the employees' salaries:
  + Salaries: 50000, 75000, 60000, 80000
  + Sum: 50000 + 75000 + 60000 + 80000 = 265000
  + Average: 265000 ÷ 4 = 66250
* The calculation highlights the use of numeric streams and built-in aggregation methods.

### **3. Dynamic Filter: Employees Above Age Threshold**

The program asks the user for an age threshold, such as 30. Based on the input, it filters out employees younger than the threshold and displays the remaining employees:

Enter age threshold:

30

Employees above age 30:

Bob - 35 years - IT - $75000.0

Charlie - 40 years - Finance - $60000.0

Diana - 32 years - IT - $80000.0

* In this example:
  + Only employees with age > 30 are included in the result.
  + "Bob," "Charlie," and "Diana" are displayed as their ages are above 30, while "Alice" (28 years) is excluded.
* The program demonstrates the flexibility of filtering conditions, allowing user-defined thresholds to fine-tune the data.

### **4. Sorted List of Employees by Salary**

The program sorts employees by their salary in ascending order and displays the sorted list:

Employees sorted by salary (ascending):

Alice - 28 years - HR - $50000.0

Charlie - 40 years - Finance - $60000.0

Bob - 35 years - IT - $75000.0

Diana - 32 years - IT - $80000.0

* Salaries are arranged from the lowest to the highest:
  + Alice: $50000
  + Charlie: $60000
  + Bob: $75000
  + Diana: $80000
* This feature showcases the use of Comparator and sorted() in streams for organizing data.