# Overview

This Java program demonstrates the use of the built-in Function interface and Java Streams to process a dataset of employees stored in a file. It reads data from a file named 'employees.txt', transforms it using streams, filters employees over a certain age, and calculates the average salary.

# Function Interface in Java

The Function interface in Java is a part of the java.util.function package. It represents a function that accepts one argument and produces a result. This interface is typically used for transforming data or extracting a specific property from an object.  
  
Syntax:  
 Function<T, R>  
 T - input type  
 R - return type  
  
In this program, a Function is defined to map a StaffMember object to a concatenated string of name and department.

Example from the code:  
Function<StaffMember, String> labelByRole = staff -> staff.getFullName() + " - " + staff.getWorkDivision();

# Java Streams

Java Streams provide a modern and powerful way to process collections of data. A stream is a sequence of elements that supports sequential and parallel operations. Streams can be used to perform map-reduce transformations on data with clarity and efficiency.  
  
Key operations used in this program:  
- map: Transforms each element using a provided function.  
- filter: Selects elements that match a predicate.  
- collect: Gathers elements into a new collection (e.g., List).  
- average: Computes the average of numeric elements.  
  
Example from the code:  
List<String> labeledEntries = teamRoster.stream()  
 .map(labelByRole)  
 .collect(Collectors.toList());  
  
This converts each StaffMember into a string of "Name - Department" and collects them into a list.

# Benefits of Using Function and Streams

- Enhances code readability and maintainability.  
- Encourages functional programming principles.  
- Allows concise transformation and filtering of data.  
- Enables parallel processing for better performance on large datasets.

# Conclusion

Using the Function interface and Java Streams significantly improves the way collections are processed in Java. It makes the code concise, expressive, and more aligned with modern programming practices. This program demonstrates their utility by applying transformations and aggregations to employee data efficiently.