JAVA8 Stream API

The most powerful feature in Java8 is introducing Java.Util.Stream package.

By Steam API, We can do **filter,map,reduce** like operations on a collection . It is very useful as many times we need to filter a collection based on some criteria.

Like find the Employee who has salary greater than 20000.

In order to do these prior to Java8 we need to iterate over the collection then check each employee’s salary. If it is greater than 20000 we need to put it in an another list. Then printout that list. This is very time consuming and unnecessary work for developer.

But in Java8 we can do it on the fly. No need to do all those priror java8 stuff.

Now we have Stream interface you can invoke stream() function or parallelStream function on Collection to get Stream Interface.

In Stream interface you can do multiple use ful operation like filter the stream based on

Criteria to do that just need to pass a **Java.util.function.Predicate** which contains the filter criteria.

As java8 also support Lambda Expression you can pass the predicate as argument of a method.

Please read my article on java lambda Expression to know about this

<http://javaonfly.blogspot.in/2016/08/java8-lambda.html>

Enough theory let do some coding

Suppose I want to filter Employees of a company based on their salary,department,sex,manager

By Java8 we do it very easily by using filter Operation

Step 1:

Create a Employee class

**package** com.example.java;

**public** **class** Employee {

String name;

String sex;

**boolean** manager;

Integer salary;

String dept;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSex() {

**return** sex;

}

**public** **void** setSex(String sex) {

**this**.sex = sex;

}

**public** **boolean** isManager() {

**return** manager;

}

**public** **void** setManager(**boolean** manager) {

**this**.manager = manager;

}

**public** Integer getSalary() {

**return** salary;

}

**public** **void** setSalary(Integer salary) {

**this**.salary = salary;

}

**public** String getDept() {

**return** dept;

}

**public** **void** setDept(String dept) {

**this**.dept = dept;

}

@Override

**public** String toString() {

**return** "Employee [name=" + name + ", sex=" + sex + ", manager="

+ manager + ", salary=" + salary + ", dept=" + dept + "]";

}

}

Step 2:

Create another class which will perform filtering operation on Employee list

**package** com.example.impl;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Predicate;

**import** java.util.stream.Stream;

**import** com.example.java.Employee;

**public** **class** EmployeeUtiltyStream {

List<Employee> list = **new** ArrayList<Employee>();

**public** EmployeeUtiltyStream()

{

init();

}

**private** **void** init()

{

Employee emp = **new** Employee();

emp.setName("Shamik Mitra");

emp.setManager(**false**);

emp.setSex("Male");

emp.setSalary(20000);

emp.setDept("Java");

Employee emp1 = **new** Employee();

emp1.setName("Aman Verma");

emp1.setManager(**true**);

emp1.setSex("Male");

emp1.setSalary(50000);

emp1.setDept("IOT");

Employee emp2 = **new** Employee();

emp2.setName("Priti Dey");

emp2.setManager(**true**);

emp2.setSex("FeMale");

emp2.setSalary(40000);

emp2.setDept("Mainframe");

Employee emp3 = **new** Employee();

emp3.setName("Ajay Ghosh");

emp3.setManager(**false**);

emp3.setSex("Male");

emp3.setSalary(30000);

emp3.setDept("Java");

list.add(emp);

list.add(emp1);

list.add(emp2);

list.add(emp3);

}

**public** **void** filterList(Predicate<Employee> predicate) {

Stream<Employee> employeeStream = list.parallelStream();

Stream<Employee> filteredEmployeeStream = employeeStream.filter(predicate);

filteredEmployeeStream.forEach(employee->System.***out***.println(employee));

}

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

EmployeeUtiltyStream employeeUtiltyStream = **new** EmployeeUtiltyStream();

employeeUtiltyStream.filterList(employee->employee.getSalary()<=40000);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

employeeUtiltyStream.filterList( employee->employee.getSex().equalsIgnoreCase("FeMale"));

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

employeeUtiltyStream.filterList( employee->employee.isManager());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

employeeUtiltyStream.filterList( employee->employee.getDept().equalsIgnoreCase("Java"));

}

}

Please look this class very carefully

I create a init method where I add Employees in a List.

Then I create a method call filterList which takes java.util.function.Predicate class as argument

**public** **void** filterList(Predicate<Employee> predicate) {

Stream<Employee> employeeStream = list.parallelStream();

Stream<Employee> filteredEmployeeStream = employeeStream.filter(predicate);

filteredEmployeeStream.forEach(employee->System.***out***.println(employee));

}

To get Stream class I invoke parallelStram method on employee list.

Then call filter method and pass the predicate.

Now look at main method

employeeUtiltyStream.filterList(employee->employee.getSalary()<=40000);

here I invoke filterList method and pass Predicate using lambda expression

in Predicate I say filter employee based on salary less than 40,000

guess what output is

Employee [name=Shamik Mitra, sex=Male, manager=false, salary=20000, dept=Java]

Employee [name=Ajay Ghosh, sex=Male, manager=false, salary=30000, dept=Java]

Employee [name=Priti Dey, sex=FeMale, manager=true, salary=40000, dept=Mainframe]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Employee [name=Priti Dey, sex=FeMale, manager=true, salary=40000, dept=Mainframe]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Employee [name=Priti Dey, sex=FeMale, manager=true, salary=40000, dept=Mainframe]

Employee [name=Aman Verma, sex=Male, manager=true, salary=50000, dept=IOT]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Employee [name=Ajay Ghosh, sex=Male, manager=false, salary=30000, dept=Java]

Employee [name=Shamik Mitra, sex=Male, manager=false, salary=20000, dept=Java]