**filter the List of Fruits in List which is start with an alphabhet.**

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** FilterDemo {

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

// **TODO** Auto-generated method stub

List<String>fruitList= Arrays.*asList*("Apples", "Apricots", "Avocados","Bananas", "Boysenberries",

"Blueberries", "Bing Cherry","Cherries", "Cantaloupe", "Crab apples","Elderberry","Eggfruit");

fruitList=*getFilterFruitOutput*(fruitList, "A");

System.***out***.println(fruitList);

}

**private** **static** List<String> getFilterFruitOutput(List<String> namesList, String filter)

{

List<String>filterfruitlist=**new** ArrayList<>();

**for**(String name:namesList)

{

**if**(name.startsWith(filter))

{

filterfruitlist.add(name);

}

}

**return** filterfruitlist;

}

}

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.stream.Collectors;

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

**public** **class** FilterStreamDemo {

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

// **TODO** Auto-generated method stub

List<String>fruitList= Arrays.*asList*("Apples", "Apricots", "Avocados","Bananas", "Boysenberries",

"Blueberries", "Bing Cherry","Cherries", "Cantaloupe", "Crab apples","Elderberry","Eggfruit");

/\*

\* Convert list to stream.

\*

\* Returns a sequential Stream with this collection as its

\* source.

\*/

Stream<String> stream = fruitList.stream();

/\*

\* filters the name, starts with "P".

\*

\* Returns a stream consisting of the elements of this stream

\* that match the given predicate.

\*/

stream = stream.filter(name -> name.startsWith("A"));

/\*

\* Collect the output and convert streams to a List.

\*

\* Performs a mutable reduction operation on the elements of

\* this stream using a Collector.

\*/

List<String> filteredNameList = stream.collect(Collectors.*toList*());

filteredNameList.forEach(System.***out***::println);

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

List<String>filterlist= fruitList.stream().filter(fruit->fruit.startsWith("A")).collect(Collectors.*toList*());

System.***out***.println(filterlist);

}

}

* filter the Student from List of Student.
* Search student by name in the list
* Search student by name and age in the list.

**public** **class** Student {

**private** **int** id;

**private** String name;

**private** **int** age;

**public** Student(**int** id, String name, **int** age)

{

**super**();

**this**.id = id;

**this**.name = name;

**this**.age = age;

}

**public** **int** getId()

{

**return** id;

}

**public** **void** setId(**int** id)

{

**this**.id = id;

}

**public** String getName()

{

**return** name;

}

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

{

**this**.name = name;

}

**public** **int** getAge()

{

**return** age;

}

**public** **void** setAge(**int** age)

{

**this**.age = age;

}

@Override

**public** String toString()

{

**return** "Student [id=" + id + ", name=" + name + ", age=" + age

+ "]";

}

}

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** StreamStudentFilterDemo {

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

// **TODO** Auto-generated method stub

List<Student>studentList=*getStudentList*();

Scanner sc=**new** Scanner(System.***in***);

**char** ch=' ';

String name="";

**int** age=0;

Student student=**null**;

**do**

{

System.***out***.println("1. Search Student By Name");

System.***out***.println("2. Search Student By Name And Age");

System.***out***.println("3. Display Student List");

System.***out***.println("Enter your choice");

**int** choice=sc.nextInt();

**switch** (choice) {

**case** 1:

System.***out***.println("Enter Search Student Name");

name=sc.next().toLowerCase();

student=*searchStudentByName*(studentList, name);

**if**(student!=**null**)

System.***out***.println(student);

**else**

System.***out***.println(name +" is not Exists in Student List");

**break**;

**case** 2:

System.***out***.println("Enter Search Student Name");

name=sc.next().toLowerCase();

System.***out***.println("Enter student age");

age=sc.nextInt();

student=*searchStudentByNameAndAge*(studentList, name,age);

**if**(student!=**null**)

System.***out***.println(student);

**else**

System.***out***.println(name +" and "+age+" is not Exists in Student List");

System.***out***.println(student);

**break**;

**case** 3:

*displayStudentList*(studentList);

**break**;

**default**:

**break**;

}

System.***out***.println("Do u want to Continue Y/N");

ch=sc.next().toLowerCase().charAt(0);

}**while**(ch=='y');

}

**private** **static** List<Student> getStudentList()

{

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

Student s1=**new** Student(101, "sachin", 23);

Student s2=**new** Student(102, "rahul", 20);

Student s3=**new** Student(103, "dinesh", 18);

Student s4=**new** Student(104, "ajay", 15);

Student s5=**new** Student(105, "virat", 20);

list.add(s1);

list.add(s2);

list.add(s3);

list.add(s4);

list.add(s5);

**return** list;

}

**private** **static** **void** displayStudentList(List<Student> studentList)

{

studentList.forEach(System.***out***::println);

}

**private** **static** Student searchStudentByName(List<Student> studentList,String studentName)

{

Student student = studentList.stream() //convert list to stream

.filter(x -> studentName.equals(x.getName()))

.findAny() // If 'findAny' then return found

.orElse(**null**); //If not found, return null

**return** student;

}

**private** **static** Student searchStudentByNameAndAge(List<Student> studentList,String name,**int** age)

{

Student student = studentList.stream() //convert list to stream

.filter(x -> name.equals(x.getName())&& age ==x.getAge())

.findAny() // If 'findAny' then return found

.orElse(**null**); //If not found, return null

**return** student;

}

}

Create an class Product which contins fillowing fields:

**private** **int** id;

**private** String name;

**private** **int** price;

**Product.java**

**public** **class** Product {

**private** **int** id;

**private** String name;

**private** **int** price;

**public** Product(**int** id, String name, **int** price)

{

**super**();

**this**.id = id;

**this**.name = name;

**this**.price = price;

}

**public** **int** getId()

{

**return** id;

}

**public** **void** setId(**int** id)

{

**this**.id = id;

}

**public** String getName()

{

**return** name;

}

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

{

**this**.name = name;

}

**public** **int** getPrice()

{

**return** price;

}

**public** **void** setPrice(**int** price)

{

**this**.price = price;

}

@Override

**public** String toString()

{

**return** "Product [id=" + id + ", name=" + name + ", price="

+ price + "]";

}

}

**Perform following operation on the product List**

1. Search Product less than Price
2. Count Product which less than given price
3. Display Product List.
4. Sum of All Product Price.
5. Display SummaryStatistics Of Product.
6. Max Price Product Details
7. in Price Product Details.
8. Display Product Name along Price.
9. Display List Of Product Name.

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.LongSummaryStatistics;

**import** java.util.Map;

**import** java.util.Scanner;

**import** java.util.stream.Collectors;

**public** **class** StreamProductFilterDemo {

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

// **TODO** Auto-generated method stub

List<Product>productList=*getProductList*();

//System.out.println(productList.stream().map(Product::getPrice).collect(Collectors.summarizingInt(mapper)));

Scanner sc=**new** Scanner(System.***in***);

**char** ch=' ';

Product product=**null**;

**int** price=0;

**do**

{

System.***out***.println("1. Search Product less than Price");

System.***out***.println("2. Count Product which less than given price ");

System.***out***.println("3. Display Product List");

System.***out***.println("4. Sum of All Product Price");

System.***out***.println("5. Display SummaryStatistics Of Product");

System.***out***.println("6. Max Price Product Details ");

System.***out***.println("7. min Price Product Details");

System.***out***.println("8. Display Product Name along Price");

System.***out***.println("9. Display List Of Product Name ");

System.***out***.println("Enter your choice");

**int** choice=sc.nextInt();

**switch** (choice) {

**case** 1:

System.***out***.println("Enter Product Price");

price=sc.nextInt();

List<Product> filterproductList=*filterProductBasedonPrice*(productList, price);

*displayProductList*(filterproductList);

**break**;

**case** 2:

System.***out***.println("Enter Product Price");

price=sc.nextInt();

**long** countOfProducts =*countfilterProductBasedonPrice*(productList, price);

System.***out***.println("Total Prodcut Price <"+price +" "+countOfProducts);

**break**;

**case** 3:

*displayProductList*(productList);

**break**;

**case** 4:

**int** total=*sumofallProductPrice*(productList);

System.***out***.println("Total Price = "+total);

**case** 5:

*summaryStatistics*(productList);

**case** 6:

Product maxPriceProduct=*findmaxPriceProduct*(productList);

System.***out***.println("Max price product = "+maxPriceProduct);

**break**;

**case** 7:

Product minPriceProduct=*findminPriceProduct*(productList);

System.***out***.println("Min price product = "+minPriceProduct);

**break**;

**case** 8:

Map<String, Integer> productPriceMap =*getProductNameAlongPrice*(productList);

**for**(Map.Entry<String, Integer> map:productPriceMap.entrySet())

{

System.***out***.println(map.getKey() +" - "+map.getValue()+ " Rs.");

}

**break**;

**case** 9:

List<String> productNameList= *getAllProductName*(productList);

System.***out***.println(productNameList);

**default**:

**break**;

}

System.***out***.println("Do u want to Continue Y/N");

ch=sc.next().toLowerCase().charAt(0);

}**while**(ch=='y');

}

**private** **static** List<Product> getProductList()

{

List<Product> productList = **new** ArrayList<Product>();

// Adding Products

productList.add(**new** Product(1, "Sony mobile", 25000));

productList.add(**new** Product(2, "Lenovo mobile", 15000));

productList.add(**new** Product(3, "Nokia mobile", 10000));

productList.add(**new** Product(4, "Samsung mobile", 40000));

productList.add(**new** Product(5, "Real Me", 50000));

**return** productList;

}

**private** **static** **void** displayProductList(List<Product> productList)

{

productList.forEach(System.***out***::println);

}

**private** **static** List<Product> filterProductBasedonPrice(List<Product> productList,Integer price)

{

List<Product> filteredProductList = productList.stream()

.filter(p -> p.getPrice() < price)// Filter the product, whose price is less than price

.collect(Collectors.*toList*()); // collecting as list

**return** filteredProductList;

}

**private** **static** **long** countfilterProductBasedonPrice(List<Product> productList,Integer price)

{

**long** countOfProducts = productList.stream()

.filter(p -> p.getPrice() < price)// Filter the product, whose price is less than price

.count() ;// Returns the count of elements in this stream.

**return** countOfProducts;

}

**private** **static** **int** sumofallProductPrice(List<Product>productList)

{

/\*

\* reduce method takes a sequence of input elements

\* and combines them into a single summary result

\* by repeated operation.

\*

\* Using reduce method we can perform Sum, min, max, average,

\* and string concatenation.

\*

\* Here we are using reduce() method, which is used to sum of

\* all the product prices.

\*/

**int** totalPrice =productList.stream().map(Product::getPrice).reduce(0,(sum,price)->sum+price);

**return** totalPrice;

}

// sum of all the product price using Collectors method & SummaryStatistics

**private** **static** **void** summaryStatistics(List<Product> productList)

{

/\*

\* Using Collectors's method to sum the prices.

\*/

LongSummaryStatistics longSummaryStatistics = productList.stream()

.collect(Collectors.*summarizingLong*((product->product.getPrice())));

System.***out***.println(longSummaryStatistics);

System.***out***.println("Count = "+longSummaryStatistics.getCount());

System.***out***.println("Sum = "+longSummaryStatistics.getSum());

System.***out***.println("Max Price = "+longSummaryStatistics.getMax());

System.***out***.println("Min Price = "+longSummaryStatistics.getMin());

System.***out***.println("Average Price = "+longSummaryStatistics.getAverage());

}

**private** **static** Product findmaxPriceProduct(List<Product> productList)

{

// max() method to get max Product price

**return** productList.stream()

.max((product1,product2)->product1.getPrice()>product2.getPrice()?1:-1)

.get();

}

**private** **static** Product findminPriceProduct(List<Product> productList)

{

// max() method to get max Product price

**return** productList.stream()

.min((product1,product2)->product1.getPrice()>product2.getPrice()?1:-1)

.get();

}

**private** **static** Map<String, Integer> getProductNameAlongPrice(List<Product> productList)

{

// Converting Product List into a Map

Map<String, Integer> productPriceMap = productList.stream()

.collect(Collectors.*toMap*(p -> p.getName(), p -> p.getPrice()));

**return** productPriceMap;

}

**private** **static** List<String> getAllProductName(List<Product>productList)

{

**return** productList.stream().map(Product::getName).collect(Collectors.*toList*());

}

}