## Create class Person as follows: import java.util.List; import java.util.ArrayList; import java.time.chrono.IsoChronology; import java.time.LocalDate;

public class Person {

public enum Sex {

MALE, FEMALE

}

String name;

LocalDate birthday;

Sex gender;

String emailAddress;

Person(String nameArg, LocalDate birthdayArg,

Sex genderArg, String emailArg) {

name = nameArg;

birthday = birthdayArg;

gender = genderArg;

emailAddress = emailArg;

}

public int getAge() {

return birthday

.until(IsoChronology.INSTANCE.dateNow())

.getYears();

}

public void printPerson() {

System.out.println(name + ", " + this.getAge());

}

public Sex getGender() {

return gender;

}

public String getName() {

return name;

}

public String getEmailAddress() {

return emailAddress;

}

public LocalDate getBirthday() {

return birthday;

}

public static int compareByAge(Person a, Person b) {

return a.birthday.compareTo(b.birthday);

}

public static List<Person> createRoster() {

List<Person> roster = new ArrayList<>();

roster.add(

new Person(

"Fred",

IsoChronology.INSTANCE.date(1980, 6, 20),

Person.Sex.MALE,

"fred@example.com"));

roster.add(

new Person(

"Jane",

IsoChronology.INSTANCE.date(1990, 7, 15),

Person.Sex.FEMALE, "jane@example.com"));

roster.add(

new Person(

"George",

IsoChronology.INSTANCE.date(1991, 8, 13),

Person.Sex.MALE, "george@example.com"));

roster.add(

new Person(

"Bob",

IsoChronology.INSTANCE.date(2000, 9, 12),

Person.Sex.MALE, "bob@example.com"));

return roster;

}

}

1. Create Test class to implement the following functionalities:

List<Person> roster = Person.createRoster();  
System.out.println("Contents of roster:");

* 1. Average age of male members  
     double average = roster.stream()  
      .filter(p -> p.getGender() == Person.Sex.MALE)  
      .mapToInt(Person::getAge)  
      .average()  
      .getAsDouble();

System.out.println("Average age (bulk data operations): " + average);

* 1. Sum of ages with sum operation  
     Integer totalAge = roster

.stream()

.mapToInt(Person::getAge)

.sum();

System.out.println("Sum of ages (sum operation): " +

totalAge);

// 3. Sum of ages with reduce(identity, accumulator)

Integer totalAgeReduce = roster

.stream()

.map(Person::getAge)

.reduce(

0,

(a, b) -> a + b);

System.out.println(

"Sum of ages with reduce(identity, accumulator): " +

totalAgeReduce);

// 4. Average of male members with collect operation

/\* Averager averageCollect = roster.stream()

.filter(p -> p.getGender() == Person.Sex.MALE)

.map(Person::getAge)

.collect(Averager::new, Averager::accept, Averager::combine);

System.out.println("Average age of male members: " +

averageCollect.average());\*/

// 5. Names of male members with collect operation

System.out.println("Names of male members with collect operation: ");

List<String> namesOfMaleMembersCollect = roster

.stream()

.filter(p -> p.getGender() == Person.Sex.MALE)

.map(p -> p.getName())

.collect(Collectors.toList());

namesOfMaleMembersCollect

.stream()

.forEach(p -> System.out.println(p));

// 6. Group members by gender

System.out.println("Members by gender:");

Map<Person.Sex, List<Person>> byGender =

roster

.stream()

.collect(

Collectors.groupingBy(Person::getGender));

List<Map.Entry<Person.Sex, List<Person>>>

byGenderList =

new ArrayList<>(byGender.entrySet());

byGenderList

.stream()

.forEach(e -> {

System.out.println("Gender: " + e.getKey());

e.getValue()

.stream()

.map(Person::getName)

.forEach(f -> System.out.println(f)); });

// 7. Group names by gender

System.out.println("Names by gender:");

Map<Person.Sex, List<String>> namesByGender =

roster

.stream()

.collect(

Collectors.groupingBy(

Person::getGender,

Collectors.mapping(

Person::getName,

Collectors.toList())));

List<Map.Entry<Person.Sex, List<String>>>

namesByGenderList =

new ArrayList<>(namesByGender.entrySet());

namesByGenderList

.stream()

.forEach(e -> {

System.out.println("Gender: " + e.getKey());

e.getValue()

.stream()

.forEach(f -> System.out.println(f)); });

// 8. Total age by gender

System.out.println("Total age by gender:");

Map<Person.Sex, Integer> totalAgeByGender =

roster

.stream()

.collect(

Collectors.groupingBy(

Person::getGender,

Collectors.reducing(

0,

Person::getAge,

Integer::sum)));

List<Map.Entry<Person.Sex, Integer>>

totalAgeByGenderList =

new ArrayList<>(totalAgeByGender.entrySet());

totalAgeByGenderList

.stream()

.forEach(e ->

System.out.println("Gender: " + e.getKey() +

", Total Age: " + e.getValue()));

// 9. Average age by gender

System.out.println("Average age by gender:");

Map<Person.Sex, Double> averageAgeByGender =

roster

.stream()

.collect(

Collectors.groupingBy(

Person::getGender,

Collectors.averagingInt(Person::getAge)));

for (Map.Entry<Person.Sex, Double> e : averageAgeByGender.entrySet()) {

System.out.println(e.getKey() + ": " + e.getValue());

}

}

}