```
package steamdemo;


import java.util.*;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class StreamDemo {
    public static void main(String[] args) {
        Person p1 = new Person("Nikit", 5);
        Person p2 = new Person("Akash", 3);
        Person p6 = new Person("Akash", 2);
        Person p3 = new Person("KKKK", 2);
        Person p4 = new Person("tttt", 4);
        Person p5 = new Person("yyyy", 1);
        Person p7 = new Person("Akash", 2);


        List<Person> personList =  List.of(p1, p2, p3, p4, p5, p6, p7);


        //output : List<Strring> which contains name
        //this map function is different from map of collection
        long startTime = System.currentTimeMillis();
        System.out.println(System.currentTimeMillis());
        Comparator<Person> nameComarator =
Comparator.comparing(Person::getName);
        Comparator<Person> ageComparator = (per1, per2) -> (per1.getAge() >
per2.getAge()) ? -1 : ((per1.getAge() == per2.getAge()) ? 0 : 1);



        List<Person> personNames = //personList.stream()
                Stream.of(p1, p2, p3, p4, p5, p6, p7)
                //.map(abc -> abc.getName())
                //.sorted(Comparator.comparing(Person::getName).thenComparing((per
1, per2) -> Integer.compare(per2.getAge(), per1.getAge())))
```

```
                .distinct()
                .collect(Collectors.toList());

        Set<String> personNameSet = //personList.stream()
                Stream.of(p1, p2, p3, p4, p5, p6, p7)
                        .map(abc -> abc.getName())
                        //.sorted(Comparator.comparing(Person::getName).thenComparing(
(per1, per2) -> Integer.compare(per2.getAge(), per1.getAge())))
                        .collect(Collectors.toSet());




        System.out.println("input : "+personList);
        System.out.println("output : "+personNames);
        System.out.println("outputSet : "+personNameSet);


    }
}

package steamdemo;

import java.util.Objects;

public class Person implements Comparable<Person>{
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }



    public String getName() {
        return name;
    }
```

```
public void setName(String name) {
    this.name = name+getPersonalId(name);
}

private String getPersonalId(String name) {
    return "testId";
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

@Override
public String toString() {
    return "Person{" +
            "name='" + name + '\'' +
            ", age=" + age +
            '}';
}

@Override
public int compareTo(Person o) {
    return this.getName().compareTo(o.getName());
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Person person = (Person) o;
    return age == person.age && Objects.equals(name, person.name);
}

@Override
```

```
    public int hashCode() {
        return Objects.hash(name, age);
    }
}

package steamdemo;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class FlatMapDemo {
    public static void main(String[] args) {
        List<String> maharashtraDistrictNames = List.of("Mumbai", "Pune",
"Nagpur");
        List<String> telenganaDistrictNames = List.of("Hyderabad",
"SomeDistrict");

        List<List<String>> stateDistrictList = List.of(maharashtraDistrictNames,
telenganaDistrictNames);

        List<String> district = stateDistrictList.stream()
                .flatMap(list -> list.stream())
                .collect(Collectors.toList());

        //Function<T,Stream> mapper

        System.out.println("input list : " + stateDistrictList);
        System.out.println("district list : " + district);



        Person p1 = new Person("Nikit", 5);
        Person p2 = new Person("Akash", 3);
        Person p6 = new Person("Akash", 2);
        Person p3 = new Person("KKKK", 2);
        Person p4 = new Person("tttt", 4);
        Person p5 = new Person("yyyy", 1);
        Person p7 = new Person("Akash", 2);
        List<Person> group1 = List.of(p1, p2, p3, p4);
        List<Person> group2 = List.of(p5, p6, p7);
        List<List<Person>> groupList = List.of(group1, group2);

        List<Person> allPerson = groupList.stream()
                .flatMap(group -> group.stream())
                .collect(Collectors.toList());

        System.out.println("Input list : "+groupList);
        System.out.println("Output list : "+allPerson);

        List<List<List<String>>> someList = new ArrayList<>();
        List<String> flattenedList = someList.stream()
                .flatMap(lst -> lst.stream()
                                .flatMap(lst2 -> lst2.stream()))
                .collect(Collectors.toList());

        List<Object> colors = List.of("Red","Yellow","White");
        List<Object> numbers = List.of(100,300,400,700,200,500);


        List<List<Object>> ColorNumber = List.of(colors,numbers);
    }
}

package steamdemo;

import java.util.List;
import java.util.Optional;

public class StreamDemo2 {
    public static void main(String[] args) {
        Person p1 = new Person("Nikit", 5);
        Person p2 = new Person("Akash", 3);
        Person p3 = new Person("AKKKK", 2);
        Person p4 = new Person("tttt", 4);
        Person p5 = new Person("yyyy", 1);
        Person p6 = new Person("Akash", 2);
        Person p7 = new Person("Akash", 2);

        List<Person> personList =  List.of(p1, p2, p3, p4, p5, p6, p7);

        Optional<Person> firstElement = personList.stream()
                .filter(person -> person.getAge()== 2)
                .findFirst();
        System.out.println(firstElement);

        Optional<Person> anyElement = personList.parallelStream()
                .filter(person -> person.getAge()== 2)
                .findAny();
        System.out.println(anyElement);

        boolean isAnyElementMatchingPredicate = personList.stream()
                .filter(person -> person.getName().startsWith("A"))
                .anyMatch(person -> person.getAge() == 2);
        System.out.println("isAnyElementMatchingPredicate : 
"+isAnyElementMatchingPredicate);

        boolean isAllElementMatchingPredicate = personList.stream()
                .filter(person -> person.getName().startsWith("A"))
                //*IMP - If stream is empty then allMatch will return true regardless of 
the predicate
                .allMatch(person -> person.getAge() == 2);
        System.out.println("isAllElementMatchingPredicate : 
"+isAllElementMatchingPredicate);

        personList.stream()
                .filter(p -> p.getName().startsWith("A"))
                .forEach(p -> System.out.println(p.getName()));
    }
}

package steamdemo;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

public class CollectorDemo {


    public static void main(String[] args) {
        Person p1 = new Person("Nikit", 5);
        Person p2 = new Person("Akash", 3);
        Person p3 = new Person("AKKKK", 2);
        Person p4 = new Person("tttt", 4);
        Person p5 = new Person("yyyy", 1);




        List<Person> personList =  List.of(p1, p2, p3, p4, p5);

        LinkedList<Person> filteredList = personList.stream()
                .filter(person -> person.getName().startsWith("A"))
                .collect(() -> new LinkedList<Person>(),
                        (list, element) -> list.add(element),
                        (list1, list2) -> list1.addAll(list2));

        //it shoudl return String with pipe '|' separated
        System.out.println(filteredList);

        //Map<Name, age>

        Map<String, Integer> personMap = personList.stream()
                //.filter(person -> person.getName().startsWith("A"))
                .collect(Collectors.toMap(person -> person.getName(), person ->
person.getAge()));

        System.out.println(personMap);

    }
}
```

Assignment question :
Create list of person,
filter the list based on name which starts with A
filter the list based on age > 18
map to name of the person
add prefix “Mr. “ to each name
print the list of name