**Exercise: Common Functional Interfaces using Lambda Expressions**

**Note: To be followed by previous 3 assignments**

Consider modified Person and PersonDetails class as follows:

public class Person {  
 private String name;  
 private int age;  
 private PersonGender gender;  
}

public class PersonDetails {  
 //Name: name  
 private String formattedName;  
 //Age: age years  
 private String formattedAge;  
 //Gender: gender  
 private String formattedGender;  
}

Given a list of People, conditions are as follows -

a.     Filter all people with gender “Male” and age greater than 20.

b.     Filter all people with gender “Female” and age between 25 and 40.

c.     Filter all people with name start with a and gender Male

**Predicate: T-> Boolean**

**Tasks:**

Write down a **PersonPredicateImpl** class with three different predicates for the required conditions above.

Write a filter method in PersonHelper.java that accepts two parameters (List of persons and a Predicate) and it returns a filtered list of persons based on the passed predicate.

Use this filter method in test class to print the list of filtered employees.

**Consumer: T -> ()**

**Tasks:**

Write down a **PersonConsumerImpl** class with three different Consumers for the required conditions above, so as to print the filtered persons.

Write a print method in PersonHelper.java that accepts two parameters (List of persons and a Consumer) and it prints a filtered list of persons based on the passed consumer.

Use this print method in test class to print the list of filtered employees.

**Supplier: () -> T**

**Tasks:**

1. Write down a **PersonSupplierImpl** with two different Suppliers for returning new Person and PersonDetails object.
2. Write an add method in PersonHelper.java that accepts two parameters (List of persons and Supplier) and it adds a new Person to the existing person list.
3. The same task as above can be carried out for PersonDetails.
4. In test class just print the Person/PersonDetails list to verify the output, or else you can check the details of last person in the list.

**Function: T -> R**

**Requirement: Display Person details.**

Client 1: Given a person, display the person details as a plain string. (toSting())

(Person -> String)

Client 2: Given a person, display the person details as per PersonDetails model.

formattedName : will hold formatted name. ex: (Name: name)

formattedAge : will hold formatted age. ex: (Age: age years)

formattedGender: will hold formatted gender ex: (Gender: gender)

**Tasks:**

1. Write down a PersonFunctionImp with two different Functions that takes

* Person as parameter and return string
* Person as parameter and return PersonDetails

1. After that, write a map method in PersonMapperHelper.java that accepts two parameters (List of persons and a Function) and it returns a filtered list of persons based on the passed Function.
2. With this contract defined, write test cases in [PersonMapperTest.java](https://github.com/MyronRogtao/java8features/blob/master/src/test/java/my/tutorials/behaviorparameterization/TransactionMapperTest.java) to test each client requirement.

**Additional Tasks:**

* Which one is best, custom or built in functional interface? When to use which one?
* Find out more built in functional interfaces, and try to implement small example programs for them.