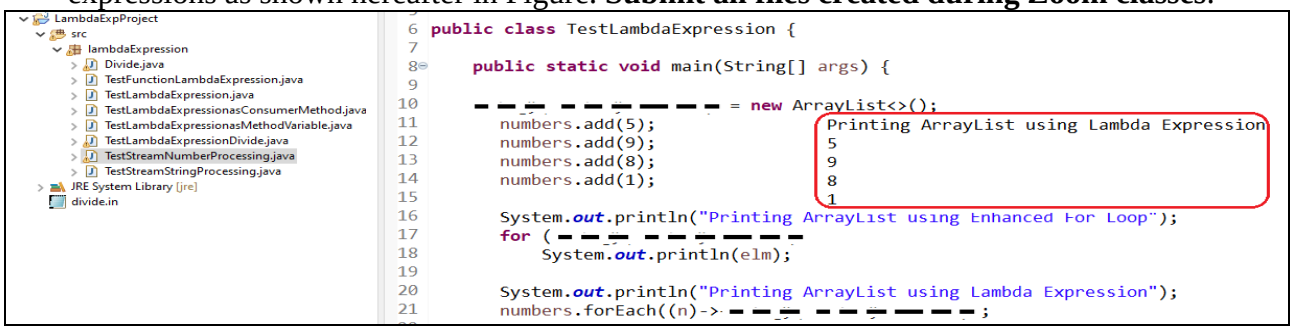
CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Programming Algorithms and Patterns
420-930-VA

Teacher: Samir Chebbine **Lab 5** **Aug 14, 2023**

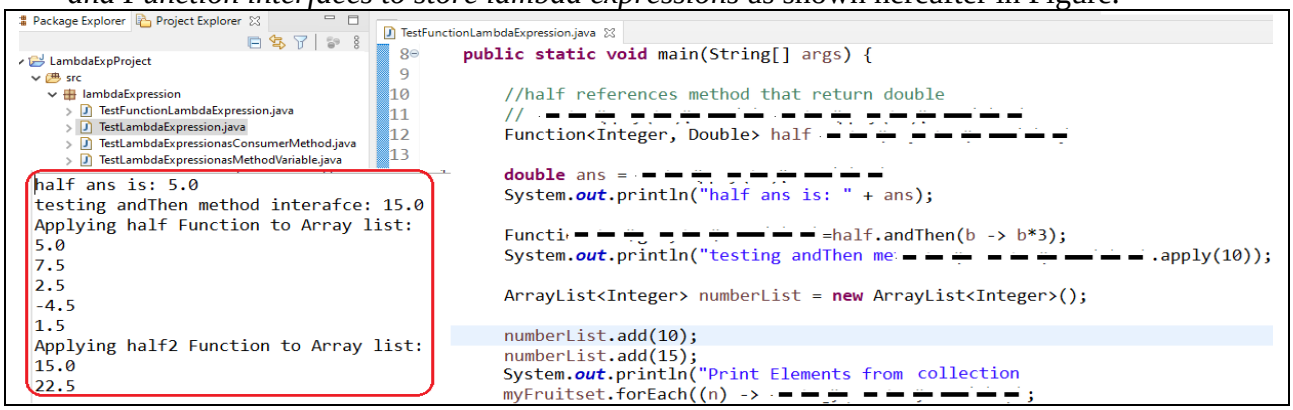
Lab 5: Lambda expressions and Stream processing

Complete all these following programs as explained during **Zoom Synchronous classes.** All *missing coding statements* were provided there with explanation. **Create and Submit** a Word file ***Lab5ProgramminAlgorithmsandPatternsYourName.docx*** which includes **output screenshots** for every Java Project. Submit Java projects too.

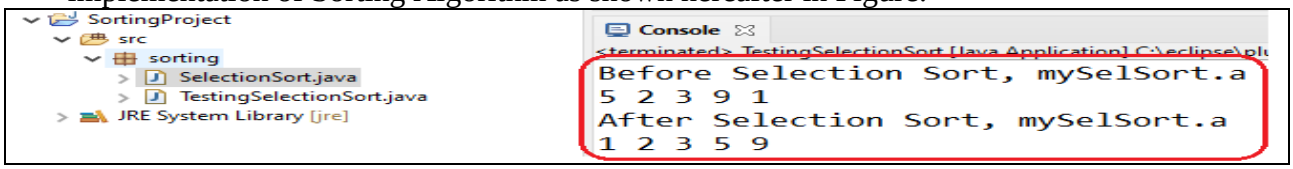
1. **Using Lambda Expressions**
   Create *LambdaExpressionProject* using Eclipse IDE for demonstrating the use of lambda expressions as shown hereafter in Figure. **Submit all files created during Zoom classes**.



2. **Consumer and Function Interface**
   Create testing Java classes as done during Zoom class to demonstrate the use of *Consumer and Function interfaces to store lambda expressions* as shown hereafter in Figure.



3. **Sorting Algorithms**
   Create Selection Sort Java class as done during Zoom class to demonstrate the implementation of Sorting Algorithm as shown hereafter in Figure.



4. Applying Lambda Expressions to HashSet collection

- Create *LambdaTripProject* as shown in Figure, to store the records of the file *Trip.in* (use delimiter \t to read *Trip.in*) onto an *HashSet* using the method `add()`.
- Create a Java class *Trip*, to define data structure type, called *Trip*, which includes the following members (the same as in Lab 4):
  a. The private data members: *emp_id* (Integer), *emp_name* (String), *emp_address* (String), *emp_gasprice* (double), *emp_distance* (int), *emp_costhotel* (double), and *emp_costfood* (double). This order represents the columns in the file *Trip.in*
  b. Add Mutator **(**setter**)** methods in `Trip` class to *modify* the values of private members.
  c. Add Accessor **(**getter**)** methods in `Trip` class to *access* the values of private members.
  d. Add a method *toString()* that prints class data attributes in the form of:
  "Emp Id = "emp_id ", Emp Name = " emp_name ", Emp Add = "emp_address ", gas_price = " *emp_gasprice* ", distance = " *emp_distance* ", cost_hotel = " *emp_costhotel* ", cost_food = " *emp_costfood*
  e. Add a method (*CalculateCostTrip()* that calculates, and returns the cost of a trip (cost trip = (*emp_distance* * *emp_gasprice*) + *emp_costhotel* + *emp_costfood*)
  f. Add a void method *printCostTrip()* that prints class data attributes in the form of
  "Emp Id= "emp_id ", Emp Name= " emp_name ", Emp Add= "emp_address ", gas_price = "*emp_gasprice* ", distance= "*emp_distance* ",cost_hotel= " *emp_costhotel*",cost_food="*emp_costfood*,"Total Cost Trip= "CalculateCostTrip()
- Add every record stored as an object into *HashSet* using the method `add(Trip wrecord)`
- Display the number of elements of the *HashSet* using the method `size()`.
- Print all elements of the *HashSet applying Lambda expression* invoking *toString*() method as shown hereafter.

```
The Employee Trip information you entered are: 6

The Employee Trip information using Lambda Expression
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0
```

Skipping through the element of the Set and printing collection using Lambda Expression invoking toString()

- Add static void method *lambda_printCostTrip* in the main testing class that calls the instance method *printCostTrip ()* defined in *Trip* class.
- Print then all elements of the *HashSet applying Lambda expression* invoking *lambda_printCostTrip* method as shown hereafter.

```
The Employee Trip information you entered are: 6

The Employee Trip information using Lambda Expression
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0

 Invoking printCostTrip method using Lambda Expression
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5, Total Cost = 128.49$
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0, Total Cost = 312.08$
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0, Total Cost = 180.50$
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0, Total Cost = 392.50$
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0, Total Cost = 612.00$
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0, Total Cost = 493.10$
```

Skipping through elements of the set and printing collection using Lambda Expression invoking *lambda_printCostTrip (Trip...)*

- Print then all elements of the *HashSet* using *method reference operator* :: invoking *printCostTrip()* instance method of the class Trip as shown hereafter.

```
The Employee Trip information you entered are: 6

The Employee Trip information using Lambda Expression
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0

 Invoking printCostTrip method using Lambda Expression
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5, Total Cost = 128.49$
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0, Total Cost = 312.08$
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0, Total Cost = 180.50$
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0, Total Cost = 392.50$
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0, Total Cost = 612.00$
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0, Total Cost = 493.10$

 Invoking printCostTrip method using :: operator within foreach
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5, Total Cost = 128.49$
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0, Total Cost = 312.08$
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0, Total Cost = 180.50$
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0, Total Cost = 392.50$
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0, Total Cost = 612.00$
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0, Total Cost = 493.10$
```

Skipping through the element of the Set and printing collection using Lambda Expression invoking toString()

Skipping through elements of the set and printing collection using Lambda Expression invoking *lambda_printCostTrip (Trip...)*

Skipping through elements of the set and printing collection using Method reference operator :: invoking printCostTrip() instance method of the class Trip

- Define in the *main* class *tripDiscount* variable of type *Function<Double, Double>* interface that stores *Lambda expression method* with one parameter, its expression returns total cost trip after applying a discount of 10% on calculated cost trip *CalculateCostTrip()*
- Apply the trip discount functional interface *tripDiscount* on all elements of the trip *HashSet* and display the new cost of trip after discount as shown hereafter.

```
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0, Total Cost = 493.10$

 Applying discount Function to Trip set using Lambda Expression:
Cost Trip after Discount for 5, Paul Tremblay is: 115.64$
Cost Trip after Discount for 1, Stev Jeff is: 280.87$
Cost Trip after Discount for 2, Amine Khan is: 162.45$
Cost Trip after Discount for 3, Eduard Becker is: 353.25$
Cost Trip after Discount for 4, James Peter is: 550.80$
Cost Trip after Discount for 6, Paul Henry is: 443.79$
```

- Define in the *main* class *tripAdvanceFee* variable of type *Function<Double, Double>* interface that stores *Lambda expression method* with one parameter, its expression returns total trip advance fee providing an advance fee of 30% to employee trip applied after Function *tripDiscount* using *andThen()* functional method.
- Apply the trip advance fee functional interface *tripAdvanceFee* on all elements of trip *HashSet*. Display new cost of trip after discount and total trip advance fee as shown here.

```
 Applying discount Function to Trip set using Lambda Expression:
Cost Trip after trip discount for 5, Paul Tremblay is: 115.64$
Cost Trip after trip discount for 1, Stev Jeff is: 280.87$
Cost Trip after trip discount for 2, Amine Khan is: 162.45$
Cost Trip after trip discount for 3, Eduard Becker is: 353.25$
Cost Trip after trip discount for 4, James Peter is: 550.80$
Cost Trip after trip discount for 6, Paul Henry is: 443.79$

 Applying tripAdvanceFee Function to Trip set using
 "andThen" method with Lambda Expression after tripDiscount :
Cost Trip advance fee for 5, Paul Tremblay is: 34.69$
Cost Trip advance fee for 1, Stev Jeff is: 84.26$
Cost Trip advance fee for 2, Amine Khan is: 48.73$
Cost Trip advance fee for 3, Eduard Becker is: 105.98$
Cost Trip advance fee for 4, James Peter is: 165.24$
Cost Trip advance fee for 6, Paul Henry is: 133.14$
```

- Define in the *main* class *totaltripCostMethod* variable of type *Consumer<…>* interface that stores *Lambda expression method* with one parameter, and invoke void method *printCostTrip()* from *Trip* class.
- Test *totaltripCostMethod* variable functional interface using its functional method *accept()* on record (2,"Amine Khan", "Paris France", 1.11, 50, 75.00, 50.00) as shown hereafter.

```
 Applying tripAdvanceFee Function to Trip set using
 "andThen" method with Lambda Expression after tripDiscount :
Cost Trip advance fee for 5, Paul Tremblay is: 34.69$
Cost Trip advance fee for 1, Stev Jeff is: 84.26$
Cost Trip advance fee for 2, Amine Khan is: 48.73$
Cost Trip advance fee for 3, Eduard Becker is: 105.98$
Cost Trip advance fee for 4, James Peter is: 165.24$
Cost Trip advance fee for 6, Paul Henry is: 133.14$

Using Consumer Functional interface
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0, Total Cost = 180.50$
```

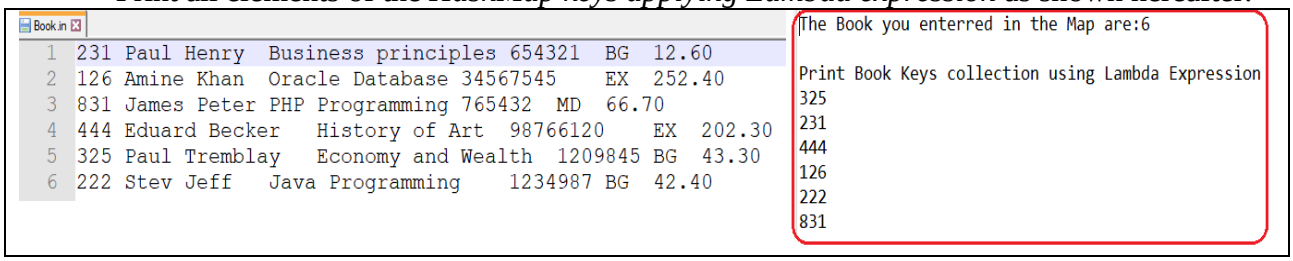
5. Applying Stream Processing to Trip HashSet collection

- Invoke *collection stream* methods in the *main* class to process elements of the *Trip HashSet* such as *filter*, *sorted*, *max*, *min*, *anyMatch*, as shown hereafter:
- Display the Number of Employees in the HashSet whose *Total Trip Cost* > 400$
- Display Employees in the HashSet sorted by *Emp_id*.
- Display Employees in the HashSet sorted by *CalculateCostTrip*.
- Display *Max Cost Trip* of Employee in the HashSet.
- Display *Min Cost Trip* of Employee in the HashSet.
- Display if Employee Trip info matching *emp_name* "Eduard" is in the HashSet.
- Display all Employee Trip info all matching *emp_name* "Paul" in the HashSet.

```
Using Stream Processing filter Method
Number of Employees in the HashSet whose Total Trip Cost > 400$ is: 2

Using Stream Processing sorted Method
Display Employees in the HashSet sorted by Emp_id:
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0

Using Stream Processing sorted Method
Display Employees in the HashSet sorted by CalculateCostTrip:
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 2, Emp Name = Amine Khan, Emp Add = Paris France, gas_price = 1.11, distance = 50, cost_hotel = 75.0, cost_food = 50.0
Emp Id = 1, Emp Name = Stev Jeff, Emp Add = 112, New York Central Park, gas_price = 1.09, distance = 112, cost_hotel = 150.0, cost_food = 40.0
Emp Id = 3, Emp Name = Eduard Becker, Emp Add = Helsinki, Sweden, gas_price = 1.01, distance = 200, cost_hotel = 110.5, cost_food = 80.0
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0

Using Stream Processing max Method
Display Max Cost Trip of Employee in the HashSet:
Emp Id = 4, Emp Name = James Peter, Emp Add = Nairobi, Kenya, gas_price = 0.99, distance = 300, cost_hotel = 245.0, cost_food = 70.0
Cost Trip: 612.0

Using Stream Processing min Method
Display Min Cost Trip of Employee in the HashSet:
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Cost Trip: 128.49

Using Stream Processing anyMatch Method
Display if Employee Trip info matching emp_name Eduard is in the HashSet:true

Display all Employee Trip info all matching emp_name Paul in the HashSet:
Emp Id = 5, Emp Name = Paul Tremblay, Emp Add = Sidney, Australia, gas_price = 1.15, distance = 20, cost_hotel = 69.99, cost_food = 35.5
Emp Id = 6, Emp Name = Paul Henry, Emp Add = Los_Anglos, USA, gas_price = 0.98, distance = 95, cost_hotel = 315.0, cost_food = 85.0
```

6. Applying Stream Processing to HashMap collection

- Create *LambdaHashMapBookProject* as shown in Figure, to store the records of the file *Book.in* (use delimiter \t to read *Book.in*) onto an *HashMap*.
- Use class Book (from Lab3/Lab4) to represent a single record (`b_id, b_author, b_title, b_isbn, b_type, b_price`).
  a. The method called public `calculate_Price_Euro()` is already implemented in `Book` class to calculate the price in euro using the following formula: `b_price*0.7`
  b. Add a method `doBookDiscount()` that calculates, and returns book price discount on book price such as: book_discount = (`b_price*0.2`)
  c. Method public String toString() in `Book` class to print the `Book` information in the form of `"Book [" + b_id + "//"+ b_author + "//" + b_title + "//" + b_isbn + "//" + b_type+ "//" + b_price + " Book Price Euro=", calculate_Price_Euro())", Book Price Discount=", doBookDiscount ())"]";`
  d. Notice that content of *Book.in* is updated from version used in Lab3/Lab4.
- Add every record stored as an object into *HashMap* using the method `put`.
- Display the number of elements of the *HashMap* using the method `size()`.
- Print all elements of the *HashMap keys applying Lambda expression* as shown hereafter.



- Print then all elements of book *HashMap applying Lambda expression* invoking toString*()* method as shown hereafter.

```
Print Book info V collection using Lambda Expression
Book [325//Paul Tremblay//Economy and Wealth//1209845//BG//43.3$,
 Book Price Euro=30.31, Book Price Discount  =8.66$]
Book [231//Paul Henry//Business principles//654321//BG//12.6$,
 Book Price Euro=8.82, Book Price Discount  =2.52$]
Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$]
Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
Book [222//Stev Jeff//Java Programming//1234987//BG//42.4$,
 Book Price Euro=29.68, Book Price Discount  =8.48$]
Book [831//James Peter//PHP Programming//765432//MD//66.7$,
 Book Price Euro=46.69, Book Price Discount  =13.34$]
```

- Print all elements of the *HashMap sorted* with respect to *key b_id* as shown hereafter.

```
--- Sorted Book Map (Sorted by Key) ---
126=Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
222=Book [222//Stev Jeff//Java Programming//1234987//BG//42.4$,
 Book Price Euro=29.68, Book Price Discount  =8.48$]
231=Book [231//Paul Henry//Business principles//654321//BG//12.6$,
 Book Price Euro=8.82, Book Price Discount  =2.52$]
325=Book [325//Paul Tremblay//Economy and Wealth//1209845//BG//43.3$,
 Book Price Euro=30.31, Book Price Discount  =8.66$]
444=Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$]
831=Book [831//James Peter//PHP Programming//765432//MD//66.7$,
 Book Price Euro=46.69, Book Price Discount  =13.34$]
```

- Print all elements of the *HashMap sorted* with respect *to value of book discount* invoking *doBookDiscount ()* as shown hereafter.

```
--- Sorted Book Map (Sorted by Value doBookDiscount) ---
231=Book [231//Paul Henry//Business principles//654321//BG//12.6$,
 Book Price Euro=8.82, Book Price Discount  =2.52$]
222=Book [222//Stev Jeff//Java Programming//1234987//BG//42.4$,
 Book Price Euro=29.68, Book Price Discount  =8.48$]
325=Book [325//Paul Tremblay//Economy and Wealth//1209845//BG//43.3$
 Book Price Euro=30.31, Book Price Discount  =8.66$]
831=Book [831//James Peter//PHP Programming//765432//MD//66.7$,
 Book Price Euro=46.69, Book Price Discount  =13.34$]
444=Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$]
126=Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
```

- Print all elements of the *HashMap sorted* with respect *to value of book price in Euro* invoking *calculate_Price_Euro ()*.
- Print all elements of the *HashMap sorted* with respect *to value of book price.*
- Print all elements of the *HashMap sorted* with respect *to value of book title* as shown hereafter.

```
--- Sorted Book Map (Sorted by Value getB_Title) ---
231=Book [231//Paul Henry//Business principles//654321//BG//12.6$,
 Book Price Euro=8.82, Book Price Discount  =2.52$]
325=Book [325//Paul Tremblay//Economy and Wealth//1209845//BG//43.3$,
 Book Price Euro=30.31, Book Price Discount  =8.66$]
444=Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$]
222=Book [222//Stev Jeff//Java Programming//1234987//BG//42.4$,
 Book Price Euro=29.68, Book Price Discount  =8.48$]
126=Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
831=Book [831//James Peter//PHP Programming//765432//MD//66.7$,
 Book Price Euro=46.69, Book Price Discount  =13.34$]
```

- Print all elements of the *HashMap sorted* with respect *to value of book Author* in *reverse order* as shown hereafter.

```
--- Sorted Book Map (Sorted by Value getB_Author) reversed ---
222=Book [222//Stev Jeff//Java Programming//1234987//BG//42.4$,
 Book Price Euro=29.68, Book Price Discount  =8.48$]
325=Book [325//Paul Tremblay//Economy and Wealth//1209845//BG//43.3$,
 Book Price Euro=30.31, Book Price Discount  =8.66$]
231=Book [231//Paul Henry//Business principles//654321//BG//12.6$,
 Book Price Euro=8.82, Book Price Discount  =2.52$]
831=Book [831//James Peter//PHP Programming//765432//MD//66.7$,
 Book Price Euro=46.69, Book Price Discount  =13.34$]
444=Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$]
126=Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
```

- Display *Max book price discount* in the book HashMap as shown hereafter.

```
Using Stream Processing max Method
Display Max Book Discount in the HashMap:
Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]
```

- Search for any matching of Book Type of "EX" in HashMap using *filter( )* as shown hereafter.

```
 Using filter() to search for any matching of Book Type of "EX" in HashMap
[444, 126]
[Book [444//Eduard Becker//History of Art//98766120//EX//202.3$,
 Book Price Euro=141.61, Book Price Discount  =40.46$], Book [126//Amine Khan//Oracle Database//34567545//EX//252.4$,
 Book Price Euro=176.68, Book Price Discount  =50.48$]]
```