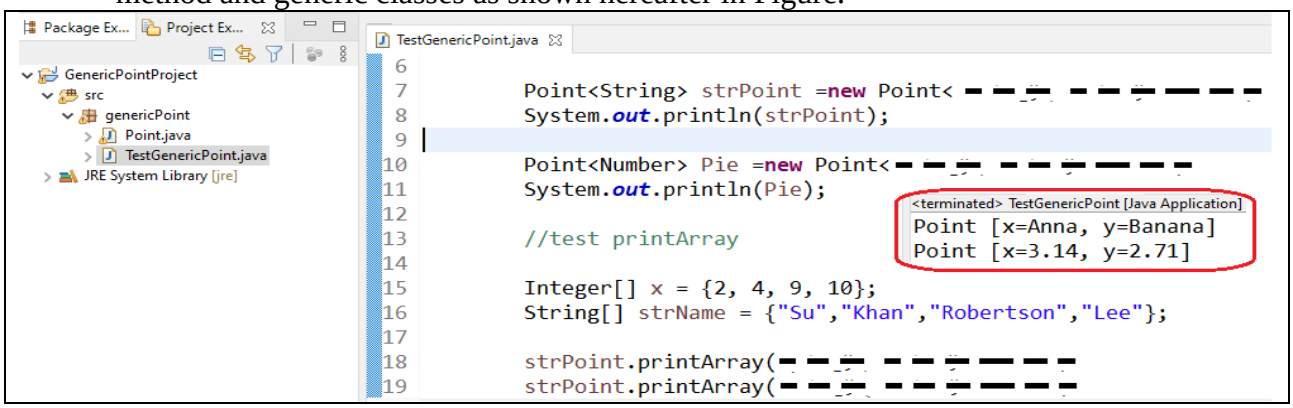
CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Programming Algorithms and Patterns
420-930-VA

Teacher: Samir Chebbine **Lab 4** **Aug 07, 2023**

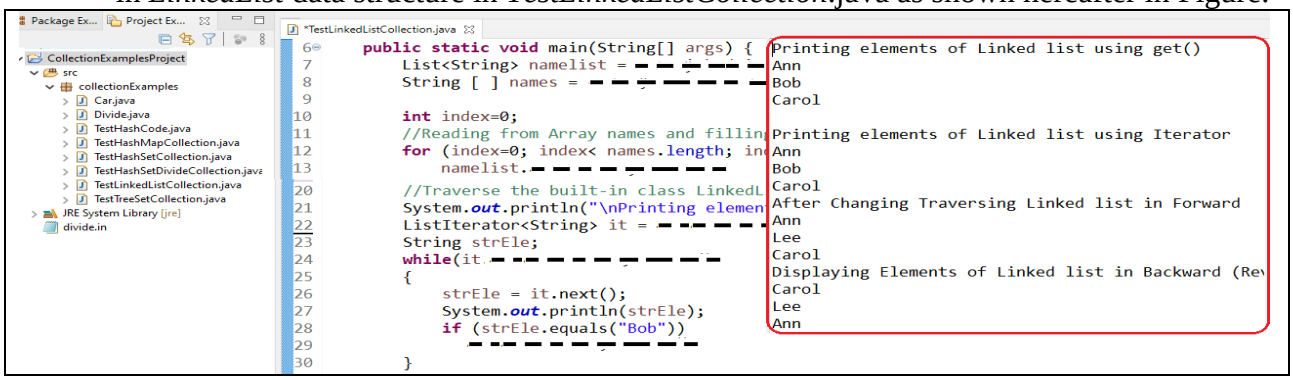
Lab 4: Generic Classes and Java Collections Framework

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 ***Lab4ProgramminAlgorithmsandPatternsYourName.docx*** which includes **output screenshots** for every Java Project. Submit the Java projects too.

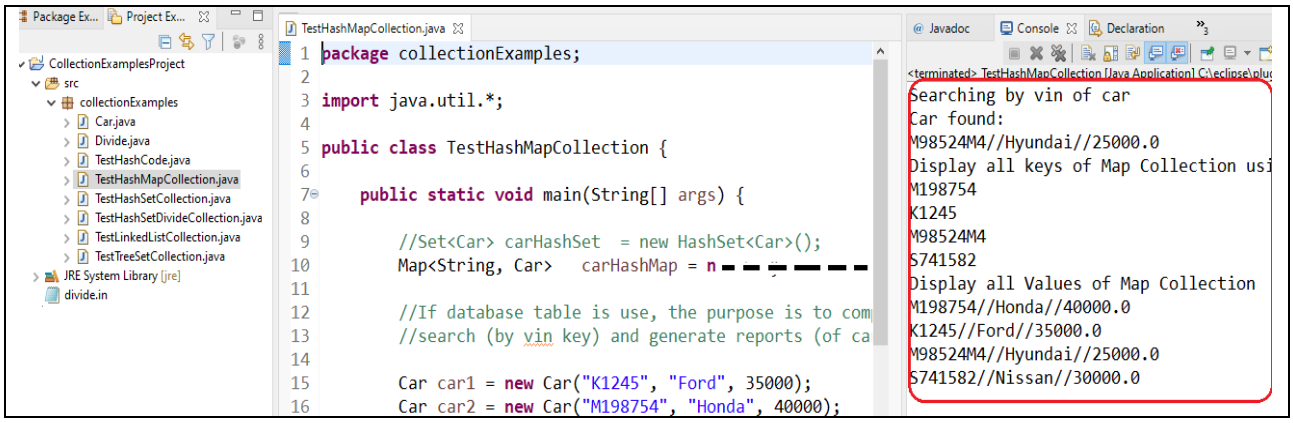
1. **Generic methods & Generic classes**
   a) Create *GenericPointProject* using Eclipse IDE for demonstrating the use of generic method and generic classes as shown hereafter in Figure.



2. **Java Collection Framework (LinkedList, ArrayList)**
   a) ListIterator interface
   Create a Java Project named *CollectionExamplesProject* to demonstrate the use of ListIterator methods (*hasNext(), next(), hasPrevious(), previous()*) when traversing built-in *LinkedList* data structure in *TestLinkedListCollection*.java as shown hereafter in Figure.



   b) TreeSet, HashSet, LinkedHashSet, TreeMap, HashMap, LinkedHashMap
   Create testing Java classes as done during Zoom class to demonstrate the use of *TreeSet, HashSet, LinkedHashSet, TreeMap, HashMap, LinkedHashMap* concrete classes as shown hereafter in Figure.



3. Using ArrayList class and its methods add(), size(), get()

- Create *BookArrayListCollectionProject* as shown in Figure 1, to store records of the file *Book.in* (use delimiter \t to read *Book.in*) onto an *ArrayList* of Book class type using the method `add()`.
- Use class Book (from Lab3) to represent a single record (`b_id ,b_author,b_title,b_isbn, b_type, b_price`).
- Display number of elements of the *ArrayList* using the method `size()`.
- Print all elements of the *ArrayList* using the method `get()`.
- Print all elements of *ArrayList* using the method `next()` of `ListIterator` interface.
- Print all elements of *ArrayList* in reverse order using the method `previous()` of `ListIterator` interface.
- Add a record (13,"Joshawa Pierre","Python","1209845","BG",99.99) into *ArrayList* at index 2 using the method `add(int index, Book wrecord)`.

Figure 1

4. Using HashSet class and its methods add(), size()

- Create *CollectionTripProject* as shown in Figure 2, 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:
  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 `Client` class to *modify* the values of private members.
  c. Add Accessor **(**getter**)** methods in `Client` class to *access* the values of private members.
  d. Add a method (*CalculateCostTrip()* that calculates, and returns the cost of a trip (cost trip = (*emp_distance* * *emp_gasprice*) + *emp_costhotel* + *emp_costfood*)
- 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.*
- Print all elements of the *HashSet* using the method `next()` of `Iterator` interface.

Figure 2

- Add a record (2,"Amine Khan", "Paris France", 1.11, 50, 75.00, 50.00) into the *HashSet* using the method `add(Trip wrecord)`.
- Explain *why* the record was added to the set despite that the set does not accept *duplicated* values as shown in Figure 2.
- Add statements to prevent end user to enter duplicated information related to Trip Employee objects when the *name of employee* is already in the set as shown in Figure 2.
- Add statements to store every record stored as an object into *LinkedHashSet* in order to display the information in the *same order* as found in the input file as shown in Figure 3.

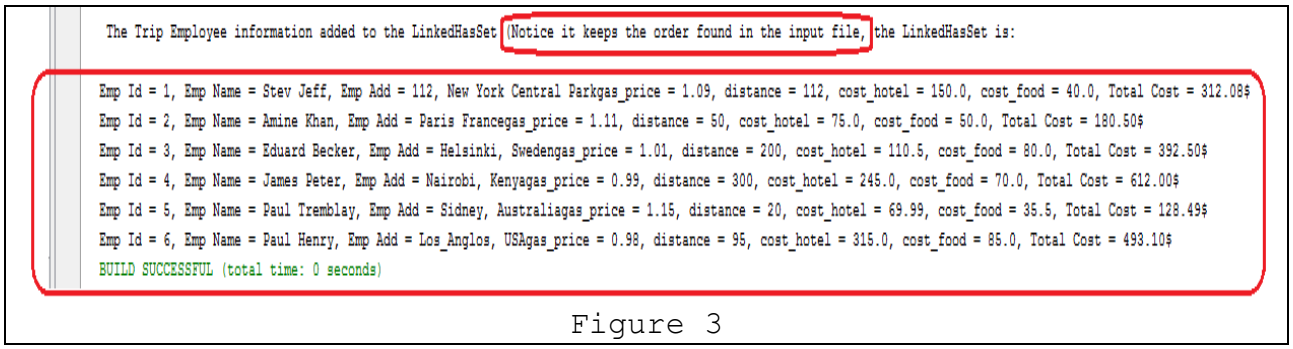


Figure 3

5. Using HashMap class and the methods put(), size(), get(), keyset(), values()

- Create *EmployeeHashMapCollectionProject* as shown in Figure 4, to store records of the file *Employee.in* onto a *HashMap* <*key, value*> using the method `put()` where key represents the *emp_id* and value represents the record of Employee class type.
- Use class Employee (from Lab3) to represent a single record (emp_id, emp_name, emp_salary).
- Display number of elements of the *HashMap* using the method `size()`.
- Search for a given emp_id into the *HashMap* entered from console and display its information using the method `get()` as shown hereafter.
- Iterate through the keys of *HashMap*, and printing each one of them.
- Iterate through the records stored in the collection and printing each one of them using `Collection` class.

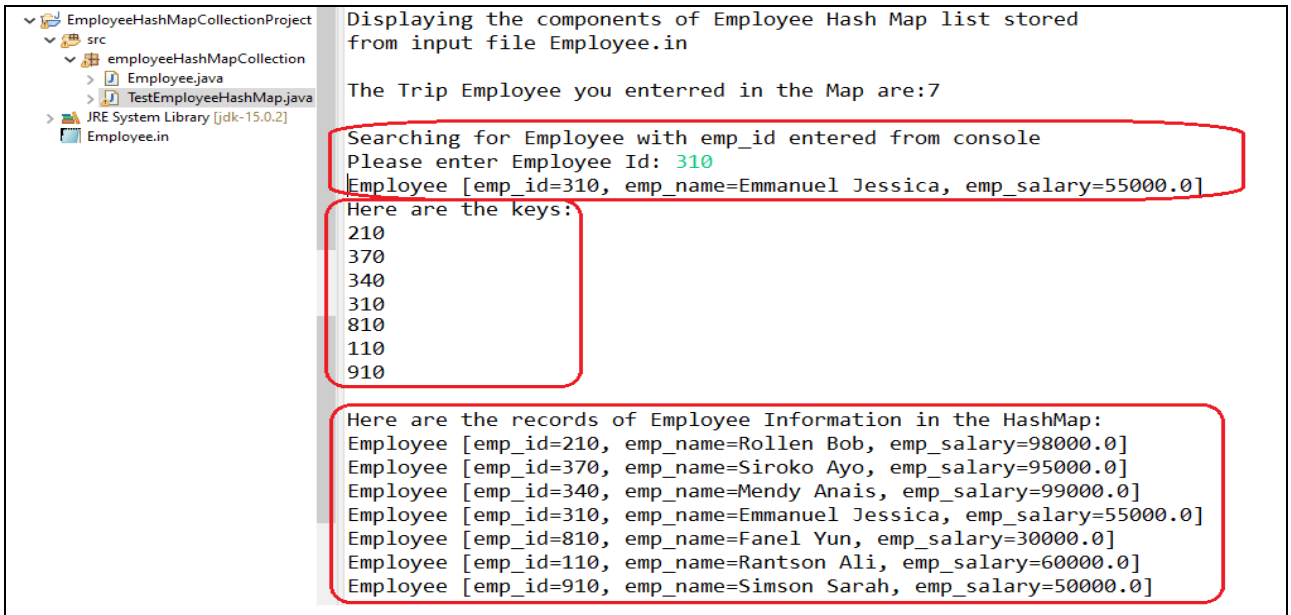


Figure 4