**LAB 9.1**

*Create a list ref with ArrayList; add some values into it and print it; see the behavior*

**LAB 9.2**

*Create a list ref with ArrayList; use its various methods (use at least 10 methods, except iterator). Aim of this assignment is to get aware with various method of List and how and when to use them*

**Steps:**

* Steps to realize / do the assignments

**TODO - Paste code snippet here**

**OUTPUT**

TODO - Paste Output of the program

**Learning:**

* Knowledge of ArrayList and its various properties / behavior

**LAB 9.3**

*Do the similar task (as Lab 9.1) for Linked list – specially use the methods which are available in LinkedList but not in ArrayList.*

**Learning:**

* Difference between ArrayList , Vector and LinkedList. Understanding of when to use which one.

**LAB 9.4**

*Do the similar task (as Lab 9.1) for HashSet and other implementing classes of Set.*

**Learning:**

* Difference between List and Set
* How and when to use HashSet

**LAB 9.4 A**

*Create a LinkedHasSet and add 4 different intereger values (say 10, 14, 12 and 8) and print this LinkedHasSet*

*Now change the LinkedHasSet into normal HashSet and execute the program again and compare the output in both the scenerio*

**Steps**

* Create a ref of Set with help of LinkedHashSet; add 4 integer values and print it
* Change the ref of Set with HashSet and execute it again

**package** cg.javaflp.practice;

**import** java.util.HashSet;

**import** java.util.LinkedHashSet;

**import** java.util.Set;

**public** **class** LinkedHashSetPractice {

**public** **static** **void** main(String[] args) {

// Set<Integer> s = new HashSet<Integer>();

Set<Integer> s = **new** LinkedHashSet<Integer>();

s.add(10);

s.add(14);

s.add(12);

s.add(8);

System.*out*.println(s);

}

}

**OUTPUT**

[10, 14, 12, 8]

**Now change the program and uncomment the commented line and comment the Line were S is referring to LinkedHashSet**

**OUTPUT**

[8, 10, 12, 14]

**Learning:**

* LinkedHashSet is an insertion ordered Set

**LAB 9.5**

*Do the similar task (as Lab 9.1) for TreeSet; try to analyze difference between HashSet and TreeSet*

**Learning:**

* What is Sorted Set and Tree Set; How and when to use

**LAB 9.5 A**

*Create a SortedSet, create two objects of StringBuffer and add them in sorted set; print size of sorted set*

**LAB 9.5 B**

*Create a class (Say AnyClass) and in main class (Having p.s.v.main method) create a sorted set and then create two objects of AnyClass and add them in sorted set, print size of set*

*Case1-*

*Now, implement comparable in AnyClass, and execute the main class again*

*Provide any implementation to compareTo method (preferable return 0)*

*Case -2*

*Now, create a subclass (say AnySubClass) which extends AnyClass; now in main method, create one object of AnyClass and one object of AnySubClass, add them to sorted set and execute the program.*

*Case3-*

Now change the compareTo of AnyClass, and instead of returning 0, return 1 and then return -1; print the sorted set of main method in all three scenario.

**Learning:**

* Only comparable object s can be added to sorted set

**LAB 9.6**

*Create a List or Set; add multiple values into it; print it with the help of Iterator.*

**Learning:**

* What is Iterator and how to use it

**LAB 9.7**

*Create a List or Set; add multiple values into it; print it with the help of enumerator and for loop (collection).*

**Learning:**

* What is enumerator and for loop for collections and how to use them

**LAB 9.8**

*Create a List or Set; add multiple String values (say names) into it; Now with the help of iterator, iterate it and append another String (say Capgemini) as surname with all the names.*

*Try to do the same with the help of enumerator and for loop.*

**Learning:**

* Difference between iterator, for loop and enumerator.
* Iterator iterate objects and may use to manipulate it; while enumerator iterate elements and for loop value of object, and they can be use only to print the value (cannot modify object)

**LAB 9.9**

*Do the similar task (as Lab 9.1) for HashMap and other implementing classes of Map.*

**Learning:**

* Understanding of Map, HashMap other implementing class of and Map; and how and when to use them. Their behavior.

**LAB 9.10**

*Create an ArrayList (without using generics) and add many values (say one integer value and all other Strings). Now in iterator cast the object value as String to print values of this ArrayList. Run the code.*

“You will find a runtime exception.”

*Now convert the same ArrayList as Generic one (Define Strings as generics values); try to do same activity as earlier.*

“You will find that system is not allowing it”

*Proceed with all String values and use iterator to iterate the list and print the values.*

“You will see no explicit casting is required”

**Learning:**

* How to use generics and why
* Benefit of generics as it convert runtime error into compile time error; also coding effort is less (do not need to cast every time)

**LAB 9.11**

*Exercise on wildcard*

**Learning:**

**LAB 9.12**

*Write a program; where end user will fill a user registration form. Create a employee POJO class (with all required properties like –email id, name, date of birth, etc – here email id must be unique); use scanner API for user interaction.*

*Now user may add/create (register) many employee; employee data must be store in a collection class.*

*System must not allow duplicate entry; to achieve this override equals method.*

*Override toString method to display customized value of employee.*

**Learning:**

* How and why to override equals and hashCode method
* Rules for equals and hashCode
* How and why to override toString method
* Use of collection objects to store data (like a runtime database)

**LAB 9.12 A**

*Write a class (say MyName ) having one global variable String name and one parameterized constructor which set value of name. Now in main class (in main method) create a Set (HashSet) and then create two objects of MyClass with same value for name field.*

*Case 1-*

*Now check the size of set*

Case 2-

Now Override hashCode and equals method of MyName for name field and execute main class again

Case 3 –

Remove hashCode and only equal method and execute it; then remove equals and keep only hasCode method and check the output

**LAB 9.13**

*Write a class, create a List and use various method of Collections Class*

**public** **class** CollectionPractice {

**public** **static** **void** main(String[] args) {

List<Integer> list = new ArrayList<Integer>();

list.add(3);

list.add(19);

list.add(12);

list.add(5);

list.add(8);

System.*out*.println("Normal: " + list);

Collections.*swap*(list, 2, 4);

System.*out*.println("Swap: " + list);

Collections.*sort*(list);

System.*out*.println("Sort: " + list);

Collections.*reverse*(list);

System.*out*.println("Sort DECD: " + list);

Collections.*shuffle*(list);

System.*out*.println("Shuffle: " + list);

List<Integer> list2 = **new** ArrayList<Integer>(5);

Collections.*copy*(list2, list);

System.*out*.println(list2);

}

}

**Learning:**

* Collections class and its usage

**LAB 9.14**

*Write a class, which have a method which adds numbers passed in its argument. Then use it for different number of arguments*

**public** **class** VariableArgument {

**public** **static** **void** main(String[] args) {

VariableArgument main = **new** VariableArgument();

**int** z = main.sumOfNumber(14, 15, 16, 44);

System.*out*.println(z);

}

**private** **int** sumOfNumber(**int**...x){

**int** z = 0;

**for** (**int** y : x){

z = z + y;

}

**return** z;

}

}