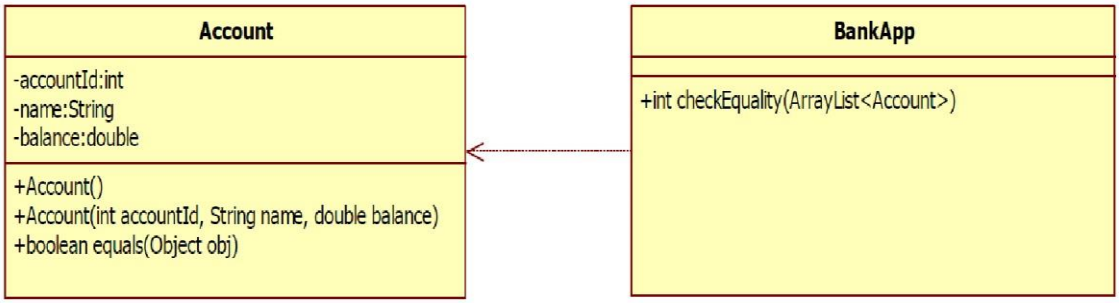
Assignment 1: Array List and equals() method

Objective:

- To understand usage of equals() method
- How to add objects to ArrayList collection
- How to retrieve objects from ArrayList collection

Problem Description:

A bank which has many customers would like to know number of customers who have same account balance. Write an application which creates below 'Account' class.



Override **"equals()"** method which compares two account balances.

Implement below method inside class called '**BankApp**', which returns number of accounts which have same account balances.

public static int checkEquality(ArrayList<Account> accountArray)

Write a test class called '**TestBank'** and implement following test cases:

Test Case	Input	Output
UTC1_01	Account account1 = new Account(1001,"Kumar",25000);	4
	Account account2 = new Account(1002,"Shanthi", 5000);	
	Account account3 = new Account(1003,"Kavya",25000);	
	Account account4 = new Account(1004,"Mohan",5000);	
	Account account5 = new Account(1005,"Dinesh",22000);	
UTC1_02	Account account1 = new Account(1001,"Kumar",25000);	2
	Account account2 = new Account(1002,"Shanthi", 5000);	
	Account account3 = new Account(1003,"Kavya",25000);	
UTC1_03	Account account1 = new Account(1001,"Kumar",23000);	0
	Account account2 = new Account(1002,"Shanthi", 6000);	
	Account account3 = new Account(1003,"Kavya",22000);	
	Account account4 = new Account(1004,"Mohan",5000);	
	Account account5 = new Account(1005,"Dinesh",29000);	

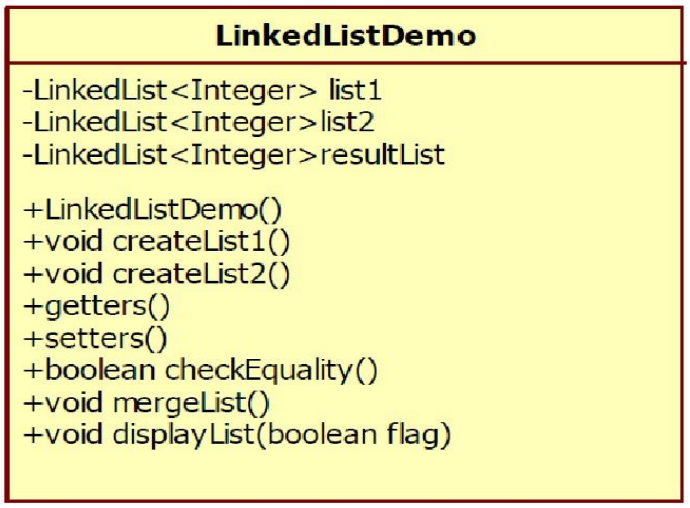
Assignment : LinkedList class and ListIterator

Objective:

- How to use of LinkedList collection
- How to compare two collection lists
- How to convert collection to arrays
- How to iterate through collections

Problem description:

Write an application to perform various operations on two Integer linked list data structures.
Below is the 'LinkedListDemo' class definition:



Following are brief description about its methods:

a) LinkedListDemo() – This constructor is used to create list1 and list2 objects of type LinkedList<Integer>

b) void createList1() – This method is used to insert few random Integer values to list1 data structure

c) void createList2() – This method is used to insert few random Integer values to list2 data structure

d) boolean checkEquality() – This method is used to check for equality of two lists and returns value **"true"** or **"false"** based on the result of comparisons

e) void mergeList() – This method is used to merge two lists if they are not equal. It should merge two lists based on unique values and stored it in result list.

f) void displayList(boolean flag) – This method used to display elements of list1 if flag is true. If flag is not true, then it must display elements of result list. It should use ListIterator to display the elements

Create class called 'TestList' and test above class methods inside main method. Here is the sample code:

```
LinkedListDemo demo = new LinkedListDemo();
demo.createList1(); // insert 10, 20, 30, 40
demo.createList2(); // insert 10, 12, 40, 50, 60
boolean flag = demo.checkEquality();

if(flag) {
    System.out.println("List1...");
    demo.displayList();
    System.out.println("List2...");
    demo.displayList(flag);
}
else {
    demo.mergeList();
    System.out.println("Resultant list...");
    demo.displayList(flag);
}
```

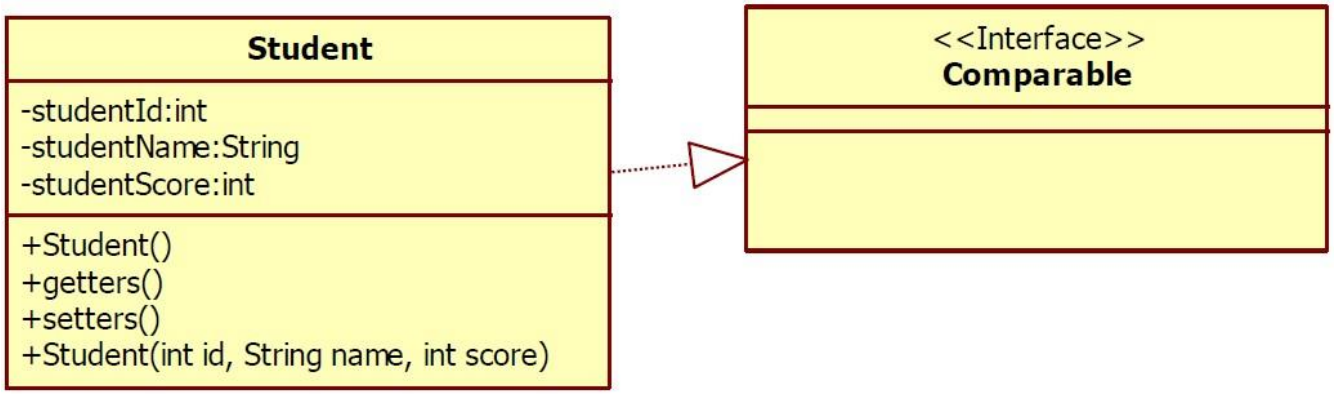
Assignment: Comparable Interface

Objective :

- How to use of comparable interface
- How to write compareTo() method

Problem description:

A college administrator who maintains student database application would like to sort list of students based on their id (ascending order). Write an application which creates following 'Student' class and sort its objects based on id.



Create a class called 'StudentAdmin' add following methods:

a) void createStudent(List<Student> list) – This method will add following Student objects to ArrayList<Student> collection

```
Student student1 = new Student(1001, "Geetha", 80);
Student student2 = new Student(1002, "Deepa", 70);
Student student3 = new Student(1003, "Shwetha", 75);
Student student4 = new Student(1004, "Lakshmi", 60);
Student student5 = new Student(1005, "Roopa", 65);
Student student6 = new Student(1006, "Mamatha", 77);
```

b) void sortStudents(List<Student> list) – This method will sort the above inserted Student's object

Create a class called 'TestStudent' and test above two methods inside main() method.

Assignment: Comparable and Comparator Interfaces

Objective:

- How to use of comparable and comparator interfaces
- How to write compareTo() method
- How to write compare method

Problem description:

Create a class called 'Employee' to hold the following data. You need to choose an appropriate collection to hold it.

Employee Id	Name	Age	Salary
17873	Meghna Tunga	21	150000
17875	Swati Suman	21	160000
17882	Subhasish	21	250000
17786	Sanket Dixit	23	200000
17007	Chandra	19	25000

Write a program to perform the following operations

- Stores the collections of the data shown above
- An operation to display all the Employees in a sorted order by default based on their employee id.
- Option for sorting based on Employee Name or Salary based on input provided at runtime.

  a. If two Employees contain same name, age should be considered.

  b. Similarly if employees have same age, salary should be considered while displaying them in order

Assignment : Set Collection - HashSet

Objective:

- How add objects to Set collection
- How to retrieve union of two sets
- How to retrieve intersection of two sets
- How to find difference from set to another

Problem description:

Write a program which performs following operations on collection of two sets of type String:

- Union
- Intersection
- Difference

Here is sample of two sets:
Set1 = [“Apple”,”Samsung”,”LG”,”Nokia”,”Micromax”];
Set2=[“Sony”,”Apple”,”Nokia”,”HTC”,”Spice”];

Your program should produce following type of output:

```
Set1=[Micromax, Apple, Nokia, LG, Samsung]
Set2=[Sony, Apple, Spice, Nokia, HTC]
Union=[Micromax, Sony, Apple, Spice, Nokia, HTC, Samsung, LG]
Intersection=[Apple, Nokia]
Difference in set1=[Micromax, Samsung, LG]
Difference in set2=[Sony, Spice, HTC]
```

Note: Use HashSet collection to perform above operations.

Assignment : Set Collection - TreeSet

Objective:

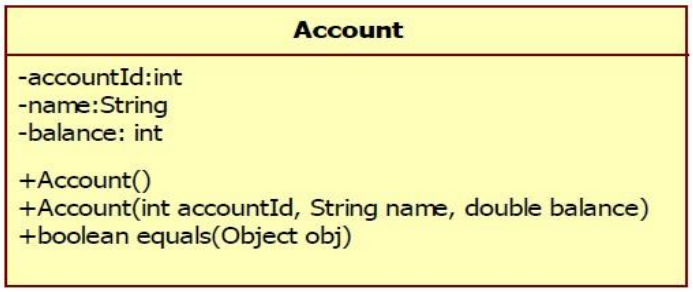
Sorting using TreeSet:

To implement your own sorting functionality with TreeSet on user defined objects, you must pass Comparator object along with TreeSet constructor call. The Comparator implementation holds the sorting logic. You have to override compare() method to provide the sorting logic on user defined objects.

Problem description:

Using TreeSet collection, sort the **"Account"** class object based on:

- Account id
- Name
- Balance



Create separate classes for each of these sorting. Call it as **"SortAccountId"**, **"SortName"** and **"SortBalance"**. In each class implement Comparator interface and override compare() method.

Create a class called **"SortAccount"** and test above sorting classes in main() method using switch case:

```
System.out.println("1.Sort Account Id");
System.out.println("2.Sort Account Name");
System.out.println("3.Sort Account Balance");
int choice=3;
switch(choice){

case 1:
        /*Write code to create and sort Account objects based on it's account Id*/
        /*Display sorted Account objects*/

        break;

case 2: /*Write code to create and sort Account objects based on it's account name*/
        /*Display sorted Account objects*/
        break;

case 3: /*Write code to create and sort Account objects based on it's account balance*/
        /*Display sorted Account objects*/

}
```

Assignment : Map collection - HashMap

Objective

- How to add objects to map collection
- How to retrieve objects from map collection

Problem description:

Write a method that takes a string and returns the number of unique characters in the string. It is expected that a string with the same character sequence may be passed several times to the method. Since the counting operation can be time consuming, the method should cache the results so that when the method is given a string previously encountered, it will simply retrieve the stored result. Use collections and maps where appropriate.

Create a class called **"StringUtility"** which has object of HashMap<String,Integer> collection, which is used to hold given string and number of unique characters in the string.

Write a method called uniqueCharacterCount(String method) which performs following operations:

a) If given string is already present in HashMap then return it's unique count

b) If given string is not present then find the number of unique characters in the string and add it to HashMap before returning it

Create a separate class called **"TestStringUtility"** and test above method in main() method:

```
public static void main(String... args) {
        StringUtility utility = new StringUtility();

        String string1 = "Pratian";
        String string2 = "Bangalore";
        String string3 = "SkillAssure";
        String string4 = "GlobalEdify";

        System.out.println(utility.uniqueCharacterCount(string1);
        System.out.println(utility.uniqueCharacterCount(string2);
        System.out.println(utility.uniqueCharacterCount(string3);
        System.out.println(utility.uniqueCharacterCount(string4);
}
```