COLLECTION FRAMEWORKS

COLLECTIONS FRAMEWORKS:-

Arrays	Collections
-Arrays are used to Store collection of homogeneous(similar) data.	-Collections is used to store Heterogeneous data as wellas Homo-geneous
-Arrays are fixed in size int a[]=new int[3];	-Collections are numerous(growable in size).
-Arrays can deals with primitive as wellas wrapper classes int a[]; Integer a[];	-Collections purely deals withWrapperclasses (Objects) ArrayList<int> a1;(Invalid) ArrayList<Integer> a1;(Valid)
-Arrays does not have any underlying data structure	-Everyclasses of Collection have datastructure
-Arrays does not contains predefine methods(add,sorting,removing,replacing) which makes manipulation of data difficult.	-In collections 80% support is by predefine API's(App programming Interface
-We should use arrays when we already knows the elements in advance	-We should go for Collection when we don't know the elements in adv.
- Memory wise array is not prefered	Memory wise collections is prefered.
-Performance wise arrays are preffered	-performance wise collection is not prefered

COLLECTION:

-collection is nothing but a group of objects represents as single unit.
-Its main purpose is to store huge amount of data.
-It provides multiple API(methods) to store and manipulate data.

//figure//

CollectionsFrameWorks:

-It provides the group of classes and interfaces which is used to store multiple objects as single unit.

collectionsFramework------->Concept
collection---------------->Root Interface
Collections--------------->class

-All the classes and interfaces of CollectionsFramework are present in java.util package.

Collection Hierarchy:-

-Collection is the root interface of CollectionFramework
-If u want to see methods of Collection interface enter cmd as
  javap java.util.Collection.

```
public interface java.util.Collection<E> extends java.lang.Iterable<E> {
  public abstract int size();
  public abstract boolean isEmpty();
  public abstract boolean contains(java.lang.Object);
  public abstract java.util.Iterator<E> iterator();
  public abstract java.lang.Object[] toArray();
  public abstract <T> T[] toArray(T[]);
  public abstract boolean add(E);
  public abstract boolean remove(java.lang.Object);
  public abstract boolean containsAll(java.util.Collection<?>);
  public abstract boolean addAll(java.util.Collection<? extends E>);
  public abstract boolean removeAll(java.util.Collection<?>);
  public boolean removeIf(java.util.function.Predicate<? super E>);
  public abstract boolean retainAll(java.util.Collection<?>);
  public abstract void clear();
  public abstract boolean equals(java.lang.Object);
  public abstract int hashCode();
  public java.util.Spliterator<E> spliterator();
  public java.util.stream.Stream<E> stream();
  public java.util.stream.Stream<E> parallelStream();
```

-Above methods are available in that collection interface
-All the classes which are implementing collection interface can use these methods
-Collection interface is extended by 3 different interfaces
1.List<Interface>
2.Set<Interface>
3.Queue<Interface>
-Learning about collection means learning about classes and its characteristics
-Different class of collection are:

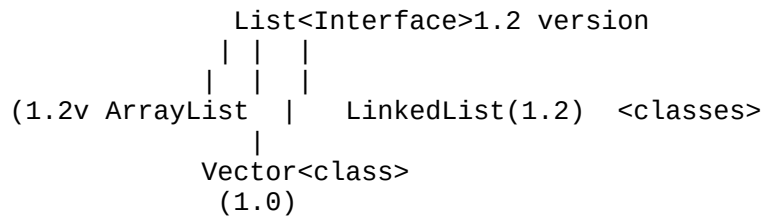
1.ArrayList
2.LinkedList
3.Vector
implements List Interface

4.PriorityQueue
5.LinkedList
implements Queue Interface

6.Treeset
7.Hashset
8.LinkedHashSet
implements SetInterface
characteristics to study for everyclass

-version
-homogeneous or heterogeneous
-Null object is possible r not
-duplicates allowed r not
-insertion order is preserve r not
-data structure
-cursors

List Interface:-



-List is an Interface which extends Collection Interface.
-List is implemented by 3 different class
1.ArrayList
2.LinkedList
3.Vector(class vector extends Stack implements List)
-javap java.util.List

```
C:\Users\User>javap java.util.List
Compiled from "List.java"
public interface java.util.List<E> extends java.util.Collection<E> {
  public abstract int size();
  public abstract boolean isEmpty();
  public abstract boolean contains(java.lang.Object);
  public abstract java.util.Iterator<E> iterator();
  public abstract java.lang.Object[] toArray();
  public abstract <T> T[] toArray(T[]);
  public abstract boolean add(E);
  public abstract boolean remove(java.lang.Object);
  public abstract boolean containsAll(java.util.Collection<?>);
  public abstract boolean addAll(java.util.Collection<? extends E>);
  public abstract boolean addAll(int, java.util.Collection<? extends E>);
  public abstract boolean removeAll(java.util.Collection<?>);
  public abstract boolean retainAll(java.util.Collection<?>);
  public void replaceAll(java.util.function.UnaryOperator<E>);
  public void sort(\java.util.Comparator<? super E>);
  public abstract void clear();
  public abstract boolean equals(java.lang.Object);
  public abstract int hashCode();
  public abstract E get(int);
  public abstract E set(int, E);
  public abstract void add(int, E);
  public abstract E remove(int);
  public abstract int indexOf(java.lang.Object);
  public abstract int lastIndexOf(java.lang.Object);
  public abstract java.util.ListIterator<E> listIterator();
  public abstract java.util.ListIterator<E> listIterator(int);
  public abstract java.util.List<E> subList(int, int);
  public java.util.Spliterator<E> spliterator();
}
```

-List follows Index Based process.
-List Allows heterogenous Objects.
-List allows Null Objects.
Useful Methods:

---

add(Object):Used to add Objects
addAll(Collection):Used to copy one collection objects into another
remove(Object):Used to Removes object
remove(Index):Used to remove the Object at given index
set(Index,Object):it replaces the old object with new Object at given index
get(index):Retrives the Object at given index
isEmpty():check whether Collectionis empty or not

contains(Object):checks whether Object is present or not
containsAll(Collection):checks whether content of one collection is present in another
collection or not.
removeAll(Collection):Rmoves whole objects of collection
size():Provides size of colletction(always calculates from 1)
Collections.sort():Sort the collection in ascending order(In case of char's as per Unicode order)
Collections.reverse():Sort in reverse way.

ArrayList :-

-ArrayList is a class which implements List Interface.
-For checking methods
 javap java.util.ArrayList
-Characteristics of ArrayList:

-It introduced in 1.2V
-ArrayList stores Heterogeneous data
-It is possible to add NULL objects in ArrayList
-It allowsDuplicateObjects
-In ArrayList Insertion Order is preserved i.e the way we added objects the way it will
 be printed.
-It follows data Structure as growable size array.
-Iterator and ListIterator cursors are used.

PROGRAM

```
package Collection.com;
import java.util.ArrayList;
public class First {
public static void main(String[] args) {
            // TODO Auto-generated method stub
            //creation of arraylist
            ArrayList a1=new ArrayList();
            //adding objects in arraylist
            a1.add("Java");
            a1.add("SQL");
            a1.add("Aptitude");
            //before 1.5
            Integer i=new Integer(100);
            a1.add(i);
            //from 1.5v autoboxing
            a1.add(300);//a1.add(Integer.valueOf(10))
            a1.add('A');//a1.add(Character.valueOf('A'))
        System.out.println(a1);//a1.toString()
        System.out.println(a1.toString());
      }

}
```

OUTPUT

```
[Java, SQL, Aptitude, 100, 300, A]
[Java, SQL, Aptitude, 100, 300, A]
```

-In Collection toString() is overridden by default i.e when we call toString() it

prints
 content of Object.
-In Collection from 1.5v whenever we add data it is being converted into Object type(Autoboxing)

Example-2

```
package Collection.com;
import java.util.ArrayList;
public class First {
public static void main(String[] args) {
            // TODO Auto-generated method stub
            //creation of arraylist
            ArrayList a1=new ArrayList();
            //adding objects in arraylist
            a1.add("Java");//0
            a1.add("SQL");//1
            a1.add("Aptitude");//2
            //before 1.5
      Integer i=new Integer(100);
            a1.add(i);
            //from 1.5v autoboxing
            a1.add(300);//a1.add(Integer.valueOf(10))
            a1.add('A');//a1.add(Character.valueOf('A'))
        System.out.println(a1);//a1.toString()
        System.out.println(a1.toString());
        //Duplicate objects are allowed
        a1.add("Java");
        a1.add(300);
        System.out.println("After adding duplicate:"+a1);
        a1.add(null);
        System.out.println("After adding NuLL Object:"+a1);
        a1.remove("Aptitude");
        a1.remove(i);
        System.out.println("After Removing few Objects:"+a1);
        a1.remove("Java");//removes first occurence of java
        System.out.println(a1);
        a1.set(0, "Aptitude");
        a1.set(2, "Selenium");
        System.out.println("After Setting some new Objects:"+a1);

      }

}
```

Output

```
[Java, SQL, Aptitude, 100, 300, A]
[Java, SQL, Aptitude, 100, 300, A]
After adding duplicate:[Java, SQL, Aptitude, 100, 300, A, Java, 300]
After adding NuLL Object:[Java, SQL, Aptitude, 100, 300, A, Java, 300, null]
After Removing few Objects:[Java, SQL, 300, A, Java, 300, null]
[SQL, 300, A, Java, 300, null]
After Setting some new Objects:[Aptitude, 300, Selenium, Java, 300, null]
```

Class-3

add(),remove()

```
package Collection.com;
```

```
import java.util.*;
public class Second {

	public static void main(String[] args) {
			// TODO Auto-generated method stub
			//creation of arraylist
			ArrayList a1=new ArrayList();
			//adding objects in arraylist
			a1.add("Java");//0
			a1.add(123);//1
			Integer i=new Integer(100);
			a1.add(i);
			a1.add(433.5f);
			a1.add("Aptitude");
			System.out.println(a1);
			a1.add(null);
			System.out.println("After adding null:"+a1);
			a1.add("Java");
			a1.add("SQL");
			System.out.println("After adding duplicates:"+a1);
			a1.remove("SQL");
			//a1.remove(123);//123 as Index
			a1.remove(1);
			a1.remove(i);
			System.out.println("After removing:"+a1);
			a1.remove("Java");//removes first java
			System.out.println("After removing duplicate:"+a1);
}
Output
------
[Java, 123, 100, 433.5, Aptitude]
After adding null:[Java, 123, 100, 433.5, Aptitude, null]
After adding duplicates:[Java, 123, 100, 433.5, Aptitude, null, Java, SQL]
After removing:[Java, 433.5, Aptitude, null, Java]
After removing duplicate:[433.5, Aptitude, null, Java]

Example-2
----------
get(),set(),for loop,size(),add(index,object)
public class Third {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	ArrayList a1=new ArrayList();
	a1.add("SQL");//0
	a1.add(100);//1
	a1.add('A');//2
	System.out.println(a1);
	System.out.println(a1.size());//3
	a1.add(0,"java");
	a1.add(1,"Apti");
	System.out.println(a1.size());//5
	System.out.println("After adding @ partpost:"+a1);
	a1.set(1,"J2EE");
	a1.set(2, "Selenium");
	System.out.println("After set()(Replacing):"+a1);
	System.out.println(a1.get(4));
	//System.out.println(a1.get(7));
	//Printing arraylist objects using get() and for loop
```

```
      System.out.println("FOrLOOP -------get()");
      for(int i=0;i<a1.size();i++)
      {
        System.out.println(a1.get(i));
      }
      }
Output
-------
[SQL, 100, A]
3
5
After adding @ partpost:[java, Apti, SQL, 100, A]
After set()(Replacing):[java, J2EE, Selenium, 100, A]
A
FOrLOOP -------get()
java
J2EE
Selenium
100
A

-Arrays are TypeSafe but collections are not type safe
Ex: int a[]=new int[3];
     a[0]=100;
     a[1]="java";//CTE
Ex:ArrayList a=new ArrayList()
   a.add("java");
   a.add(100);
   a.add('A');

Genrics
---------
-Since Collections are not type safe i.e programmer can added any types of objects
   there is no restriction for that So,To Overcome this draawback JAVA has given
Generics
concept where u can create an Object of specific type,if we add any other type of
object
than specified we will get CTE

1. ArrayList<Integer> a1=new ArrayList<>();
 a1.add("java");//CTE
 a1.add(100);
 a1.add('A');//CTE

2. ArrayList<String> a1=new ArrayList<>();
 a1.add("java");
 a1.add(100);//CTE
 a1.add('A');//CTE

3.ArrayList<Character> a1=new ArrayList<>();
 a1.add("java");//CTE
 a1.add(100);//CTE
 a1.add('A');

Example
--------
package Collection.com;
import java.util.*;
public class Forth {
```

```
     public static void main(String[] args) {
           // TODO Auto-generated method stub
 ArrayList<Integer> a1=new ArrayList<>();//only we can add integer objects
 a1.add(100);
 a1.add(200);
 a1.add(300);
 //a1.add("java");
  System.out.println("Integer arraylist is:"+a1);//[100,200,300]
  System.out.println(a1.contains(250));//false
  ArrayList<Object>  a2=new ArrayList<>();//we can add any type of objects
  a2.add("java");
  a2.addAll(a1);//To copy one  a1 arraylist into  a2 arrylist
  System.out.println("New ArrayList is:"+a2);//[java,100,200,300]
  a2.removeAll(a1);//From a2 remove all a1 objects
  System.out.println(a1.isEmpty());//false
  System.out.println(a2);//[java]
  System.out.println(a1);//[100,200,300]
  a1.removeAll(a1);//From a1 remove all the objects of a1
  System.out.println(a1.isEmpty()+"because A1 arraylist is: "+a1);

Output
------
Integer arraylist is:[100, 200, 300]
false
New ArrayList is:[java, 100, 200, 300]
false
[java]
[100, 200, 300]
truebecause A1 arraylist is:
[]


foreach syntax
--------------
for(arraytype varname:arrayname)  or  for(Collectiontype varname:Collectionname)
{
  SOP(varname);}
program
-------
package Collection.com;
import java.util.ArrayList;
import java.util.Collections;
public class SortPrt {
public static void main(String[] args) {
            // TODO Auto-generated method stub
             ArrayList<String> a21=new ArrayList<>();
              a21.add("Apti");
              a21.add("Z");
              a21.add("Sql");
              a21.add("sql");
                  Collections.sort(a21);//it is method of collections class
               //which sort the list in unicode order i.e lesser value comes
first(ascending)
                  System.out.println(a21);//[Apti,Sql,Z,sql]
                  Collections.reverse(a21);//it is method of collections class
             //which sorts the list in descending unicode order
                  System.out.println(a21);//[sql,Z,Sql,Aptitude]
                  //printing using for each loop
```

```
                for(String data:a21)
                {
                      System.out.println(data);
                }
      }


Output
------
[Apti, Sql, Z, sql]
[sql, Z, Sql, Apti]
sql
Z
Sql
Apti

Assignment on arrayList
-------------------------
1)
-Create an  ArrayList of Employee type and add 3 objects which includes
  empname,empid,empdesg

Emp class structure is
----------------------
class Emp
{
  String name,desg;
  int eid;
  Emp(String name,String desg,int id)
  {
     this.name=name;
     this.desg=desg;
     this.id=is;
  }
}

Note:   While retriving the Objects make use of instanceOf operator
-------

2)Create an ArrayList of Object type and add FOllowing type objects
   Student
   Employee
   Animal
Employee class
----------------
Refer question no-1
Student class
--------------
class Student
{
  String name,
  int id;
  Student(String name,int id)
  {
     this.name=name;
     this.id=is;
  }
}
Animal class
```

```
-------------
class Animal
{
  String name;
  String color;
  Animal(String name,String color)
  {
     this.name=name;
     this.color=color;
  }
}
```

Note:

-While retriving objects from ArrayList use for each loop and check Object types
 using instanceof operator and downcast it (For clarity refer generalisation and
specialisation
  through classes where we use instanceof operator for Animal and dog classes)

-BeginnersBook.com

-Additional ques

1.Differences between List,Set and Queue

Class-3
------
add(),remove()

```
package Collection.com;
import java.util.*;
public class Second {

      public static void main(String[] args) {
                  // TODO Auto-generated method stub
                  //creation of arraylist
                  ArrayList a1=new ArrayList();
                  //adding objects in arraylist
                  a1.add("Java");//0
                  a1.add(123);//1
                  Integer i=new Integer(100);
                  a1.add(i);
                  a1.add(433.5f);
                  a1.add("Aptitude");
                  System.out.println(a1);
                  a1.add(null);
                  System.out.println("After adding null:"+a1);
                  a1.add("Java");
                  a1.add("SQL");
                  System.out.println("After adding duplicates:"+a1);
                  a1.remove("SQL");
                  //a1.remove(123);//123 as Index
                  a1.remove(1);
                  a1.remove(i);
                  System.out.println("After removing:"+a1);
                  a1.remove("Java");//removes first java
                  System.out.println("After removing duplicate:"+a1);
```

```
}
Output
------
[Java, 123, 100, 433.5, Aptitude]
After adding null:[Java, 123, 100, 433.5, Aptitude, null]
After adding duplicates:[Java, 123, 100, 433.5, Aptitude, null, Java, SQL]
After removing:[Java, 433.5, Aptitude, null, Java]
After removing duplicate:[433.5, Aptitude, null, Java]

Example-2
----------
get(),set(),for loop,size(),add(index,object)
public class Third {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
      ArrayList a1=new ArrayList();
      a1.add("SQL");//0
      a1.add(100);//1
      a1.add('A');//2
      System.out.println(a1);
      System.out.println(a1.size());//3
      a1.add(0,"java");
      a1.add(1,"Apti");
      System.out.println(a1.size());//5
      System.out.println("After adding @ partpost:"+a1);
      a1.set(1,"J2EE");
      a1.set(2, "Selenium");
      System.out.println("After set()(Replacing):"+a1);
      System.out.println(a1.get(4));
      //System.out.println(a1.get(7));
      //Printing arraylist objects using get() and for loop
      System.out.println("FOrLOOP -------get()");
      for(int i=0;i<a1.size();i++)
      {
        System.out.println(a1.get(i));
      }
      }
Output
-------
[SQL, 100, A]
3
5
After adding @ partpost:[java, Apti, SQL, 100, A]
After set()(Replacing):[java, J2EE, Selenium, 100, A]
A
FOrLOOP -------get()
java
J2EE
Selenium
100
A

-Arrays are TypeSafe but collections are not type safe
Ex: int a[]=new int[3];
     a[0]=100;
     a[1]="java";//CTE
Ex:ArrayList a=new ArrayList()
   a.add("java");
```

```
   a.add(100);
   a.add('A');

Genrics
---------
-Since Collections are not type safe i.e programmer can added any types of objects
   there is no restriction for that So,To Overcome this draawback JAVA has given
Generics
concept where u can create an Object of specific type,if we add any other type of
object
than specified we will get CTE

1. ArrayList<Integer> a1=new ArrayList<>();
 a1.add("java");//CTE
 a1.add(100);
 a1.add('A');//CTE

2. ArrayList<String> a1=new ArrayList<>();
 a1.add("java");
 a1.add(100);//CTE
 a1.add('A');//CTE

3.ArrayList<Character> a1=new ArrayList<>();
 a1.add("java");//CTE
 a1.add(100);//CTE
 a1.add('A');

Example
--------
package Collection.com;
import java.util.*;
public class Forth {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
 ArrayList<Integer> a1=new ArrayList<>();//only we can add integer objects
 a1.add(100);
 a1.add(200);
 a1.add(300);
 //a1.add("java");
  System.out.println("Integer arraylist is:"+a1);//[100,200,300]
  System.out.println(a1.contains(250));//false
  ArrayList<Object>  a2=new ArrayList<>();//we can any type of objects
  a2.add("java");
  a2.addAll(a1);//To copy one  a1 arraylist into  a2 arrylist
  System.out.println("New ArrayList is:"+a2);//[java,100,200,300]
  a2.removeAll(a1);//From a2 remove all a1 objects
  System.out.println(a1.isEmpty());//false
  System.out.println(a2);//[java]
  System.out.println(a1);//[100,200,300]
  a1.removeAll(a1);
  System.out.println(a1.isEmpty()+"because A1 arraylist is: "+a1);

Output
------
Integer arraylist is:[100, 200, 300]
false
New ArrayList is:[java, 100, 200, 300]
false
```

```
[java]
[100, 200, 300]
truebecause A1 arraylist is: []


foreach syntax
--------------
for(arraytype varname:arrayname)  or  for(Collectiontype varname:Collectionname)
{
  SOP(varname);}
program
-------
package Collection.com;
import java.util.ArrayList;
import java.util.Collections;
public class SortPrt {
public static void main(String[] args) {
            // TODO Auto-generated method stub
             ArrayList<String> a21=new ArrayList<>();
              a21.add("Apti");
              a21.add("Z");
              a21.add("Sql");
              a21.add("sql");
                  Collections.sort(a21);//it is method of collections class
              //which sort the list in unicode order ascending
                  System.out.println(a21);//[Apti,Sql,Z,sql]
                  Collections.reverse(a21);//it is method of collections class
             //which sorts the list in descending unicode order
                  System.out.println(a21);//[sql,Z,Sql,Aptitude]
                  //printing using for each loop
                  for(String data:a21)
                  {
                        System.out.println(data);
                  }
      }


Output
------
[Apti, Sql, Z, sql]
[sql, Z, Sql, Apti]
sql
Z
Sql
AptiCursors
--------
-This is the special characteristics given for collection concepts
-Using cursors we can retrive the objects in collection one by one.

2 types
--------
1.Iterator

2.ListIterator

Iterator
---------
-Iterator is an interface which is used to traverse the list in forward direction
```

-Basically it provides the privilige to access the objects without using index.

```
interface Iterator
{
  public boolean hasNext();
  public object next();
  public void remove();
}
```

hasNext():It returns true if there is Object available
next():it returns currentObject and move the cursor to next Object

program

```
package Collection.com;
import java.util.*;
public class Std {
public static void main(String[] args) {
            // TODO Auto-generated method stub
        ArrayList<String> a1=new ArrayList<>();
        a1.add("John");
        a1.add("Riya");
        a1.add("Pooja");
        a1.add("Rohan");
        //for using iterator create its object
        Iterator<String> i1=a1.iterator();//Iterator i1=new Iterator()
        //Iterator is an interface and its object cannot be created but every
        //classes of collection contains iterator() which gives object of
        //Iterator interface
        while(i1.hasNext())
        {
            System.out.println(i1.next());//John//Riya//Pooja//Rohan
        }
      }
```

output:-

```
John
Riya
Pooja
Rohan
```

List Iterator

-List Iterator is an interafce which provides the facility to traverse the list in forward as well as backward direction.

```
interface ListIterator
{
public boolean hasNext();
public object next();
public void remove();
public boolean hasPrevious();
public object previous();
public void add();
}
```

hasprevious():

-Returns true if object is available to iterate from previous direction

previous():

-prints the current object and move cursor to next object in previous direction
hasNext():

It returns true if there is Object available
next():
----
it returns currentObject and move the cursor to next Object
add() and remove()

-For adding and removing while iterating respectively
PROGRAM

```
package Collection.com;
import java.util.ArrayList;
import java.util.ListIterator;
public class Std1 {
public static void main(String[] args) {
            // TODO Auto-generated method stub
             ArrayList<String> a1=new ArrayList<>();
              a1.add("John");
              a1.add("Riya");
              a1.add("Pooja");
              a1.add("Rohan");
              //for traversing using list iterator cursor
              ListIterator<String> l1=a1.listIterator();
              //for traversing in forward
              System.out.println("------Forward direction-------");
              while(l1.hasNext())
              {
                  System.out.println(l1.next());
              }
              //for traversing in reverse direction
              System.out.println("------Reverse direction-------");
              while(l1.hasPrevious())
              {
                  System.out.println(l1.previous());
              }
      }
```

OUTPUT :-

```
------Forward direction-------
John
Riya
Pooja
Rohan
------Reverse direction-------
Rohan
Pooja
Riya
John
```

Assignment on ArrayList

-Create an array list to add all subject of 4th year as Objects and
  iterate it using for loop

-Create the ArrayList and add any 7 colors as Objects and traverse it using

  1.forloop

```
  2.foreachloop
  3.Iterator
  4.ListIterator


-Create an array list and add 5 students as objects and create one more array list
  and add 5 more studentsas objets and copy first arraylist into second arraylist
  and
1.sort it in ascending order
2.print alternate objects
3.remove even objects

instanceOf operator:
--------------------
-   Referencevar  instanceOf classname/interfacename

-We are checking whether the reference var on left is an instance/Object of
 specified type(class or subclass  or interface type) on the right.

    Animal a1=new Animal();
    System.out.println(a1 instanceof Animal);//true

-An object of subclass type is also a type of parent class.

    class Animal
    {}
    class Dog extends Animal
    {
    }
    class C
    {PSVM()
        {
          Dog d1=new Dog();
          System.out.println(d1 instanceof Dog);//true
         System.out.println(d1 instanceof Animal);//true
          System.out.println(d1 instanceof Student);//false
       }}

   Ex3:
        class A
        {
         public static void main(String args[])
         {
             A a1=new A();
             a1=null;//abandon object
             System.out.println(a1 instanceof A);//false
          }
}

Assignment-Solutions
-------------
Student.java
-------------
package Collection.com;

public class Student {

         String name;
       int id;
```

```
        Student(String name,int id)
        {
            this.name=name;
            this.id=id;
        }
      }
```

First.java

```
------------
package Collection.com;
import java.util.ArrayList;
public class First {
public static void main(String[] args) {
            // TODO Auto-generated method stub
            //creation of arraylist
            ArrayList<Student> a1=new ArrayList<>();

            a1.add(new Student("john",4545));
            a1.add(new Student("Riya",5656));
            a1.add(new Student("Poooja",6464));
            System.out.println(a1);
            for(Student var:a1)
            {
                  if(var instanceof Student)
                  {
                        Student s2=(Student) var;
                        System.out.println(var.name+" "+var.id);
                  }
            }
      }
```

Output

```
-------
[Collection.com.Student@15db9742, Collection.com.Student@6d06d69c,
Collection.com.Student@7852e922]
john 4545
Riya 5656
Poooja 6464
```

Solution-2

```
-----------
```

Student.java

```
-------------
package Collection.com;

public class Student {

         String name;
       int id;

        Student(String name,int id)
        {
            this.name=name;
            this.id=id;
        }
      }
```

Employee.java

```
--------------
```

```
package Collection.com;

public class Employee {
   String name;
   String desg;
      public Employee(String name,String desg) {
            this.name=name;
            this.desg=desg;
      }}
Animal.java
-------------
package Collection.com;

public class Animal {

       String name;
      public Animal(String name) {
            this.name=name;
            // TODO Auto-generated constructor stub
      }

}
Sol.java
-----------
package Collection.com;
import java.util.ArrayList;
public class Sol {
public static void main(String[] args)
{
ArrayList<Object> a1=new ArrayList<>();

            a1.add(new Student("john",4545));
            a1.add(new Employee("Riya","Developer"));
            a1.add(new Animal("Tiger"));
            System.out.println(a1);
            for(Object var:a1)
            {
                  if(var instanceof Student)
                  {
                        Student s2=(Student) var;
                        System.out.println("Studentname: "+s2.name+" "+"ID:
"+s2.id);
                  }
                  else if(var instanceof Employee)
                  {
                        Employee e2=(Employee) var;
                        System.out.println("Employeename: "+e2.name+" Employee Id:
"+e2.desg);
                  }
                  else  if(var instanceof Animal)
                  {
                        Animal a2=(Animal) var;
                        System.out.println("Animal :"+a2.name);
                  }
            }
      }


      }
```

Output

[Collection.com.Student@15db9742, Collection.com.Employee@6d06d69c,
Collection.com.Animal@7852e922]
Studentname: john ID: 4545
Employeename: Riya Employee Id: Developer
Animal :Tiger

Queue interface:-

-QUeue is an Interface which is implemented by 2 different classes
 1.PriorityQueue
 2.LinkedList
-Queue is a data structure which usually follows FIFO principle(First In First Out

        Image:

PriorityQueue

-Introduced in 1.2
-It allows Only homogenous objects
-BAasically data structure of Queue is FIFO but in priority queue
 internal sorting will happen
-Duplicate objects are allowed
-Null objects are not allowed
-For retriving of data they have given the special methods called as
 peek() and poll().
-peek():It retrives the head element without deleting it
       -It returns null if there is no head element present

-poll():It retrives head element(First element) and delete it.
program

```
package Collection.com;
import java.util.*;
public class Queuingdemo {
public static void main(String[] args) {
            // TODO Auto-generated method stub
     PriorityQueue<Object> q1=new PriorityQueue<>();
      q1.add("aptitude");
      q1.add("ADP");
      q1.add("dell");
      q1.add("Bangalore");
       //Only homogeneous objects are allowed
     // q1.add(334);//classCastException
      //q1.add('A');//classCastException    Internally
      //q1.add(null);//NullPointerException<---------->q1.add(String obj=null)
      System.out.println(q1.peek());//ADP
      System.out.println(q1.poll());//ADP
      System.out.println(q1.contains("ADP"));
      while(q1.peek()!=null)
      {
        System.out.println(q1.poll());
      }
      System.out.println(q1);
      }
```

Output:

```
--------
ADP
ADP
false
Bangalore
aptitude
dell
[]
```

LinkedList

```
-----------
Linkedlist is a clss which implements List as well as Queue interface
 class LinkedList implements List,Queue

-Queue q1=new LinkedList();
-here q1 will exhibits behaviour of queue
```

Note:

```
------
-All features are same only thing is In Linkedlist implementing Queue interface
 Objects will not be sorted rather insertion order is maintained.
```

```
package Collection.com;
import java.util.*;
public class Queuingdemo {
public static void main(String[] args) {
            // TODO Auto-generated method stub
     Queue<Object> q1=new LinkedList<>();
      q1.add("aptitude");
      q1.add("ADP");
      q1.add("dell");
      q1.add("Bangalore");
      System.out.println(q1);
      //[aptitude,ADP,dell,Bangalore]
    q1.add(334);
      q1.add('A');
      System.out.println(q1.peek());//aptitude
      System.out.println(q1.poll());//aptitude
      System.out.println(q1.contains("aptitude"));
      while(q1.peek()!=null)
      {
        System.out.println(q1.poll());//ADP,dell,Bangalore,334,A
      }
      System.out.println(q1);
      }
```

Output

```
--------
[aptitude, ADP, dell, Bangalore, 123, A]
aptitude
aptitude
ADP
dell
Bangalore
123
A
[]
```

Set Interface

```
-----------------
-Set is an interface which is implemented by 3 different classes
1.Hashset
2.Treeset
3.LinkedHashset

Methods of set interface
---------------------------
javap java.util.set
public interface java.util.Set<E> extends java.util.Collection<E> {
  public abstract int size();
  public abstract boolean isEmpty();
  public abstract boolean contains(java.lang.Object);
  public abstract java.util.Iterator<E> iterator();
  public abstract java.lang.Object[] toArray();
  public abstract <T> T[] toArray(T[]);
  public abstract boolean add(E);
  public abstract boolean remove(java.lang.Object);
  public abstract boolean containsAll(java.util.Collection<?>);
  public abstract boolean addAll(java.util.Collection<? extends E>);
  public abstract boolean retainAll(java.util.Collection<?>);
  public abstract boolean removeAll(java.util.Collection<?>);
  public abstract void clear();
  public abstract boolean equals(java.lang.Object);
  public abstract int hashCode();
  public java.util.Spliterator<E> spliterator();

HashSET:-
---------
-It is a class which implements set interface
Features
-------------
-Introduced in 1.2v
-Heterogenous objects are allowed
-Duplicates are not allowed .In case if i add i won't get compile time error
 it will just add once
-only one NULL object is allowed
-Data structure is hashtable.
-Insertion order is preserved(depends on hashcode)
-Set is only uni directional so it supports only Iterator.




Program
--------
package Collection.com;
import java.util.*;
public class HAshdemo {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
      HashSet<Object> s1=new HashSet<>();
      s1.add("Rohan");
      s1.add("Riya");
      s1.add("Pooja");
      s1.add(445);
      s1.add('A');
```

```
      System.out.println("Hash set objectss are: "+s1);
      s1.add("Riya");
      s1.add(445);
      System.out.println("After add duplicates: "+s1);
      s1.add(null);
      s1.add(null);
      System.out.println("After adding null objects: "+s1);
      System.out.println("-------Forward direction-----");
      Iterator<Object> i1=s1.iterator();
      while(i1.hasNext())
      {
      System.out.println(i1.next());
      }
      }
Output
-------
Hash set objectss are: [A, Riya, Pooja, 445, Rohan]
After add duplicates: [A, Riya, Pooja, 445, Rohan]
After adding null objects: [null, A, Riya, Pooja, 445, Rohan]
-------Forward direction-----
null
A
Riya
Pooja
445
Rohan
Linked HAshSet
--------------
It is a class which extends HashSet and implements set interface
Features
-------------
-Introduced in 1.2v
-Heterogenous objects are allowed
-Duplicates are not allowed .In case if i add i won't get compile time error
 it will just add once
-only one NULL object is allowed
-Data structure is Linkedlist.
-Insertion order is preserved.
-Set is only uni directional so it supports only Iterator

//Same program only difference is output is as per insertion order//

Treeset
-------
It is aclass which implements set interafce.
Features
-------------
-Introduced in 1.2v
-Heterogenous objects are  not allowed if we add we will get class cast exception
-Duplicates are not allowed .In case if i add i won't get compile time error
 it will just add once
-No NULL object is allowed
-Data structure is tree.
-Output is in Sorted Order.
-Set is only uni directional so it supports only Iterator
Program:
--------
package Collection.com;
import java.util.*;
```

```
public class Treedemo {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	 TreeSet<Object> s1=new TreeSet<>();
	 s1.add("Java");
	 s1.add("SQL");
	 s1.add("APtitude");
	 s1.add("J2EE");
	 System.out.println("Treeset Objects are: "+s1);
	 //s1.add(122);
	 //s1.add('A');
	 //s1.add(new Student("John",566));
	 //System.out.println("Tree set after adding heterogenous obj :"+s1);
	 //s1.add(null);
	 //System.out.println("After adding null :"+s1);
	System.out.println( s1.first());
	 System.out.println(s1.last());
	 System.out.println(s1.pollFirst());
	 System.out.println(s1.pollLast());
	 System.out.println(s1.contains("APtitude"));
	 System.out.println(s1.contains("SQL"));
	}

}
Output
-------
Treeset Objects are: [APtitude, J2EE, Java, SQL]
APtitude
SQL
APtitude
SQL
false
false

-In Treeset output will defaultly in sorted order
-It allows only homogenous objects i.e it allows only comparable type of
  objects if we add heterogeneous objects it gives ClassCastExc
-If we add null objects it gives NullPointerException
-first()-->provides first object
-last()--->provides last object
-pollFirst()-->provides first object and delete it from tree
-pollLast()--->provides last object and delete it from tree

Differences between Treeset and HashSet??
Differences between Treeset and LinkedHAshSet?
dIFFERENCES BETWEEN LIST,SET,QUEUE

FOR REFERENCE
--------------
BegineersBook.com
```