Arraylist

The ArrayList class

- **ArrayList** supports dynamic arrays that can grow as needed.
- Arraylist can have duplicate elements.
- An **ArrayList** is a variable-length array of object references.
- An **ArrayList** can dynamically increase or decrease in size.
- Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array can be shrunk.

Dynamic Allocation of ArrayList

The following is a very basic idea explaining the working of the array when the array becomes full and if we try to add an item:

- Creates a bigger-sized memory on heap memory (for example memory of double size).
- Copies the current memory elements to the new memory.
- The new item is added now as there is bigger memory available now.
- Delete the old memory.

Implementation

ArrayList inherits AbstractList class and implements the List interface.

Declaring an ArrayList

ArrayList<data_type> list_name = new ArrayList<>();

- `ArrayList<Integer> arr = new ArrayList<>();`
- `ArrayList<String> arr2 = new ArrayList<>();`

Operations performed in ArrayList

- Adding element to List/ Add element
- Changing elements/ Set element
- Removing elements/Delete element
- Iterating elements
- get elements
- add elements in between two number
- Sorting elements
- ArrayList size

Add an element

```
 ArrayList <String> list = new ArrayList<>();
// Adding items to arraylist

list.add("String 1");
list.add("String 2");
list.add("String 3");
```

Get an element

- // Access items at a specific index
- String s1 = list.get(0); // s1 = "String 1"
- String s2 = list.get(2); // s2 = "String 3"

Find index of an element

- int index = list.indexOf(“string 1”);
- Return the index of first element for multiple occurences
- If not present in the Arraylist, returns -1

Remove an element

- // Remove specific item of an index
- list.remove(2);

Search an element

- List.contains(“string 3”);
- //return true if contains otherwise returns false

More operations

- // Add to specific index

```
list.add(1, "String 4");
```

- // Change item at specific index

```
list.set(1, "String 5");
```

- // Size of arraylist

```
int size = list.size();
```

- // Clear the list

```
list.clear();
System.out.println(list.size()); // prints 0
```

Sort an Arraylist

- Collections.sort(list)
- Sort list alphabetically or numerically in ascending order

Sort an Arraylist in descending order

- `Collections.sort(list,Collections.reverseOrder());`

ArrayList of different type of objects

```
ArrayList x=new ArrayList<>();
x.add("apple");
x.add(1);
x.add('3');
System.out.println(x);
if(x.contains("1")){
    System.out.println("contains");//doesn't print
}
```

Hashset

- A HashSet is a collection of items where every item is unique

Hashset

```
HashSet<Integer> a=new HashSet<>();
a.add(33);
a.add(11);
a.add(33);
System.out.println(a);
a.remove(1);
for(int i:a) System.out.println(i);
System.out.println("size of hashset:"+a.size());
if(a.contains(3)){
    System.out.println("contains");
}
```

ArrayList of Class objects

Consider Student class

```
public class student {
    public int id;
    public String name;
    public double cgpa;
    student(int i, String n, double c){
        name=n;
        id=i;
        cgpa=c;
    }
    public String toString(){
        return "name:"+name+" id:"+id+" cgpa:"+cgpa;
    }
}
```

Create ArrayList of students

```
class studentArrayList{
    public static void main(String[] args) {
        ArrayList<student> studentList=new ArrayList<>();
        studentList.add(new student(3,"Mina",3.85));
        studentList.add(new student(6,"Abir",3.25));
        studentList.add(new student(4,"Rina",3.75));
        for(student s:studentList) System.out.println(s);
        Collections.sort(studentList);//error

    }
}
```

Sort ArrayList of class objects according to name(string) in ascending order

```
Collections.sort(studentList, new Comparator<student>() {

    public int compare(student o1, student o2) {
        return o1.name.compareTo(o2.name);
    }
});
```

Sort ArrayList of class objects according to id(int) in ascending order

```
Collections.sort(studentList, new Comparator<student>() {

    public int compare(student o1, student o2) {
        return o1.id -o2.id;
    }
});
```

Sort ArrayList of class objects according to cgpa(double) in ascending order

```
Collections.sort(studentList, new Comparator<student>()
{

    public int compare(student o1, student o2) {
        if(o1.cgpa > o2.cgpa)
            return 1;
        return -1;
    }
});
```

Output:
name: Abir id:6 cgpa:3.25
name: Rina id:4 cgpa:3.75
name: Mina id:3 cgpa:3.85

Sort ArrayList of class objects according to cgpa(double) in descending order

```
Collections.sort(studentList, new Comparator<student>()
{

    public int compare(student o1, student o2) {
        if(o2.cgpa > o1.cgpa)
            return 1;
        return -1;
    }
});
```

Output:

```
name: Mina id:3 cgpa:3.85
name: Rina id:4 cgpa:3.75
name: Abir id:6 cgpa:3.25
```