Container Classes

Collection, List, Set

COMP2603
Object Oriented Programming 1

Week 8

Outline

- Java Collections Framework
- Collection Interface
  - List
    - Linked List
  - Set
    - SortedSet

Reading: Chapter 13 - Container Classes: P. Mohan 2013

Application Programming Interface (API)

What is an Application Programming Interface (API)?

https://www.youtube.com/watch?v=s7wmiS2mSXY

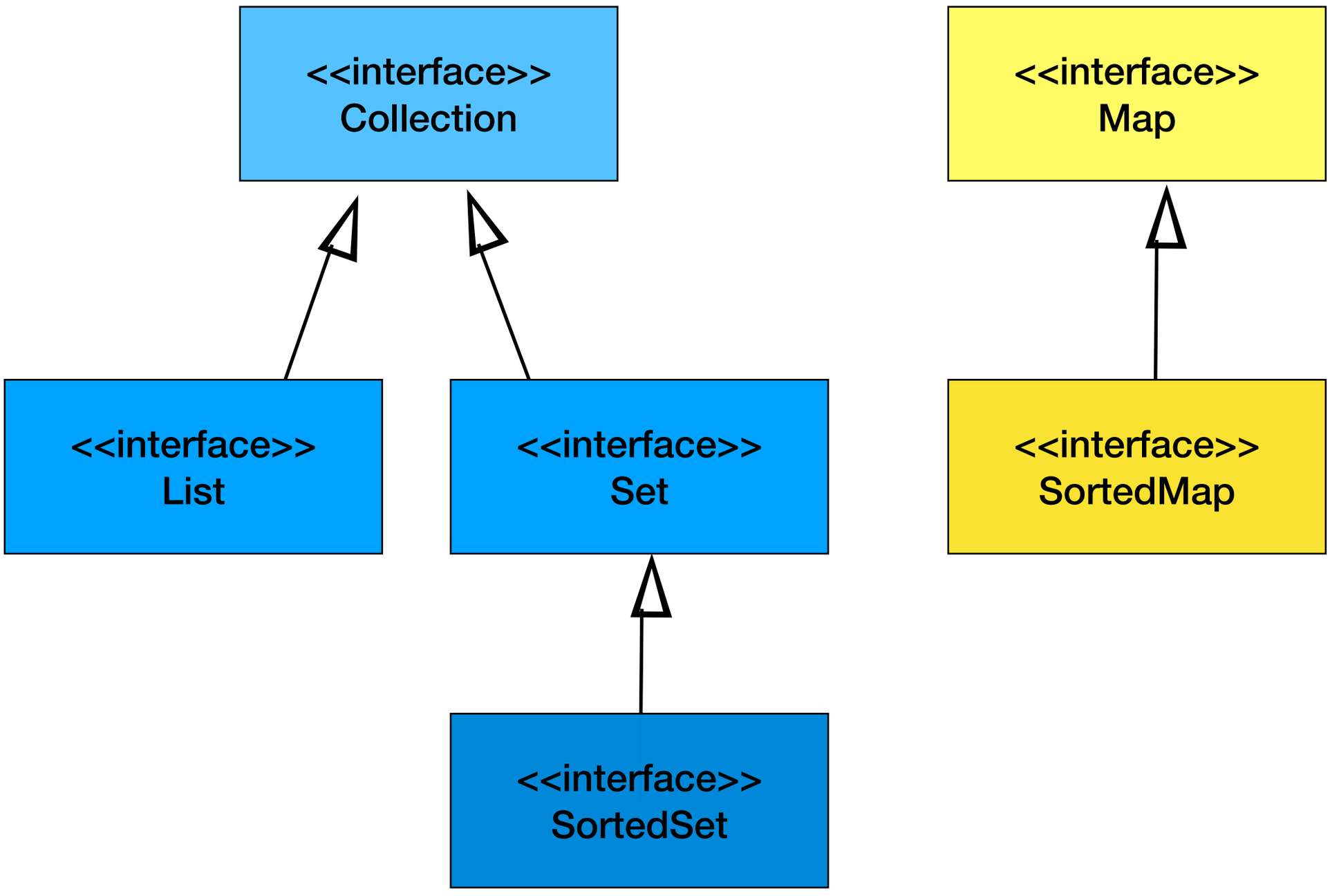
Java Collections Framework

The **Java Collections** framework contains a set of interfaces such as:

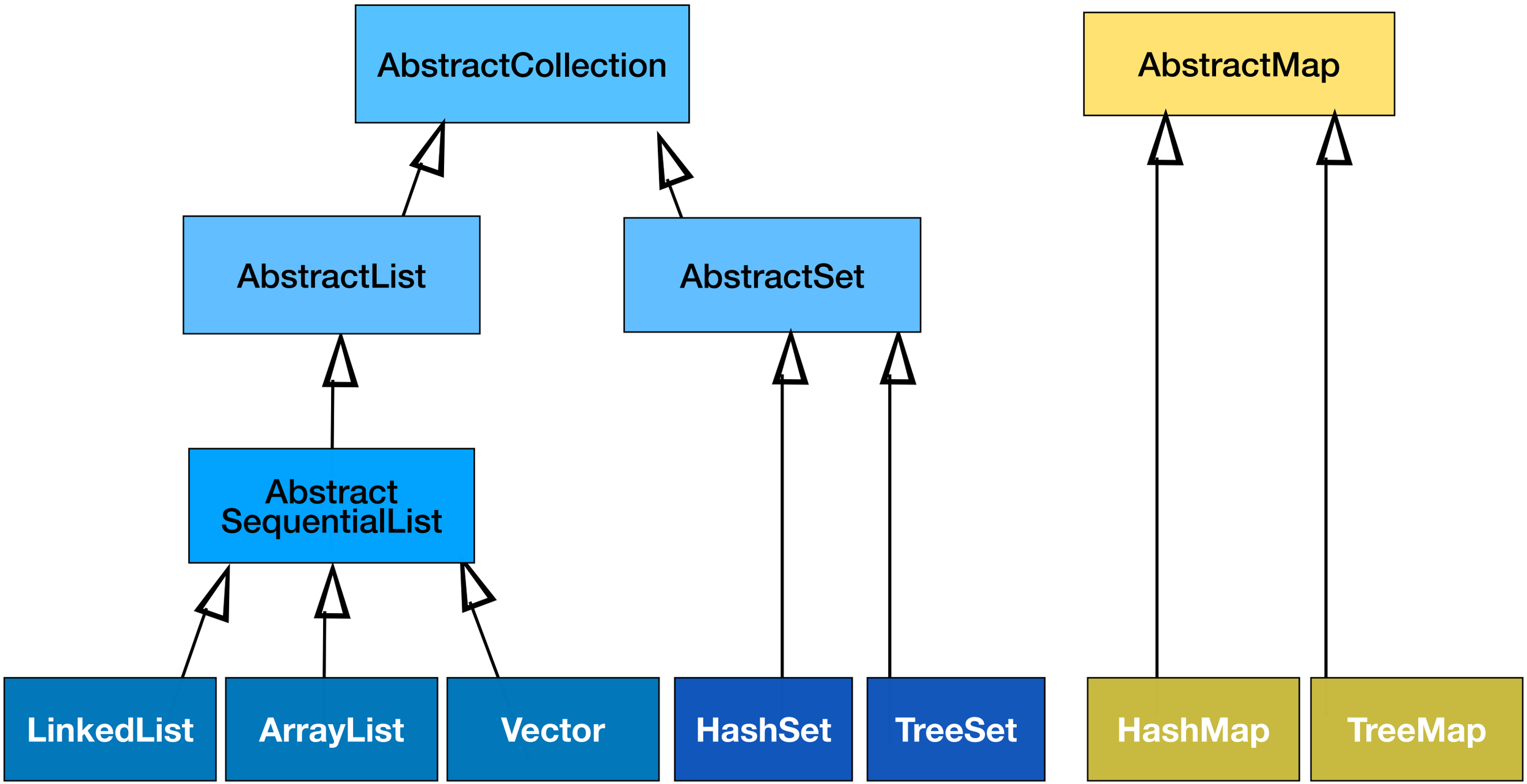
- **Collection**
- **Set**
- **List**
- **Map**.

These are used as the basis for creating concrete collection classes such as **ArrayList**, **LinkedList**, **HashSet**, and **TreeMap**.

Interfaces in the Java Collections Framework



Classes in the Java Collections Framework



The Collection Interface

The **Collection** interface represents a group or collection of objects. The objects may or may not be ordered, and the collection may or may not contain duplicate objects.

The **Collection** interface is not usually implemented directly.

Most concrete collection classes implement one or more of the specific sub-interfaces.

The Collection Interface

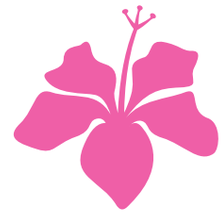
Method	Description
boolean add (E o)	Inserts the object of the specified type into the collection; returns true if the object was added, false otherwise
boolean addAll (Collection c)	Inserts all the objects from the specified collection into the current collection
void clear( )	Removes all the elements from the collection
boolean contains (Object o)	Returns true id the specified object is present in the colleciton, and false otherwise
boolean isEmpty( )	Returns true if there are no elements in the collection, and false otherwise
boolean remove(Object o)	Deletes the specified object from the collection
int size( )	Returns the number of elements currently in the collection

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Simple Plant Class

```
public class Plant{
      private String name;
      public Plant(String name){
        this.name = name
      }
      public String toString(){
        return name;
      }
 }
```

```
Plant p = new Plant(“Hibiscus”);
System.out.println(p.toString());
```



Outcome: Hibiscus

Simple Plant Class

```
public class Plant{
      private String name;
      public Plant(String name){
        this.name = name
      }
      public String toString(){
        return name;
      }
}
```

```
Plant p = new Plant("Hibiscus");
System.out.println(p);
```
// Same as printing p.toString( )

Outcome: Hibiscus

Example 0

```
Collection<Plant> plants;
//Assume:
// (1) plants collection is initialised properly
// (2) plants collection filled with 5 Plant objects
```

```
int numPlants = plants.size();
System.out.println(numPlants);
```

Outcome: 5

Example 0.1

```
Collection<Plant> plants;
//Assume:
// (1) plants collection is initialised properly
// (2) plants collection filled with 5 Plant objects
```

```
int numPlants = plants.size();
plants.clear();
System.out.println(numPlants);
```

Outcome: **5**

Example 0.2

```
Collection<Plant> plants;
//Assume:
// (1) plants collection is initialised properly
// (2) plants collection filled with 5 Plant objects
```

```
plants.clear();
System.out.println(plants.size());
```

Outcome: **0**

Example 0.3

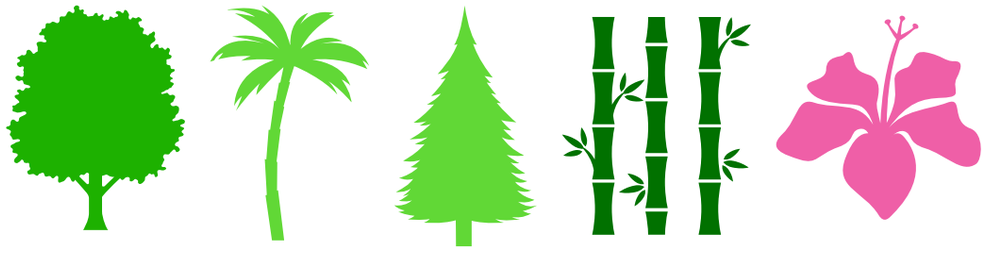
```
Collection<Plant> plants;
//plants is initialised, filled with 5 Plant objects
int numPlants = plants.size();
for( int i = 0; i < numPlants; i++){
    Plant p = plants.get(i);  // fails..why?
  System.out.println(p);
 }
```

Outcome:

Example 1

```
Collection<Plant> plants;
//Assume:
// (1) plants collection is initialised properly
// (2) plants collection filled with 5 Plant objects

int numPlants = plants.size();
 for(Plant p: plants){
     System.out.println(p);
   }
```

Outcome:
Poui
Coconut
Pine
Bamboo
Hibiscus



The Collection Interface

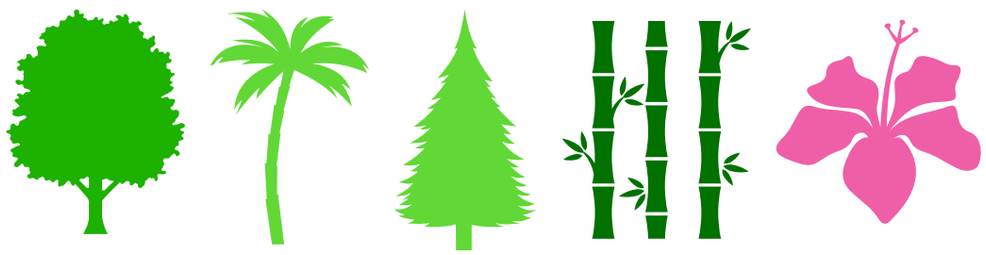
Method	Description
boolean add (E o)	Inserts the object of the specified type into the collection; returns true if the object was added, false otherwise
boolean addAll (Collection c)	Inserts all the objects from the specified collection into the current collection
void clear( )	Removes all the elements from the collection
boolean contains (Object o)	Returns true id the specified object is present in the collection, and false otherwise
boolean isEmpty( )	Returns true if there are no elements in the collection, and false otherwise
boolean remove(Object o)	Deletes the specified object from the collection
int size( )	Returns the number of elements currently in the collection

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Example 2

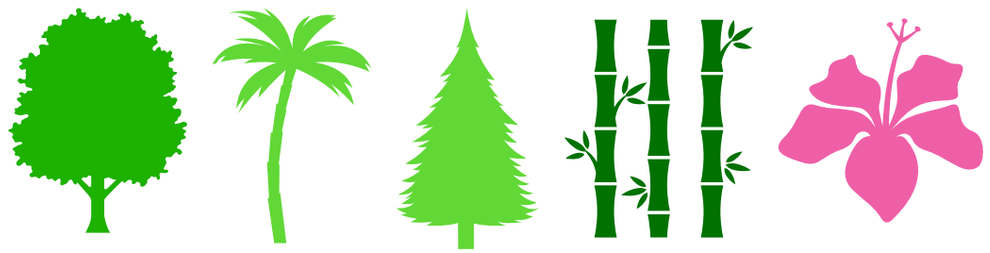
```
Collection<Plant> plants;
//plants is initialised and filled with Plant objects
Plant hp = new Plant(“Hibiscus”);
if(plants.contains(hp)){
    plants.remove(hp);
}
```

Outcome:
Poui
Coconut
Pine
Bamboo
Hibiscus **<-Not removed..why?**



Example 2

```
Collection<Plant> plants;
//plants is initialised and filled with Plant objects
Plant hp = new Plant(“Hibiscus”);     //Plant@56a081b3
if(plants.contains(hp)){
    plants.remove(hp);
}
```



Plant@36d3b4a2
Plant@26df8cb7
Plant@47b4f4f9
Plant@fbe288c
Plant@c401eb8

Outcome:
Poui
Coconut
Pine
Bamboo
Hibiscus **<-Not removed..why?**

Example 2

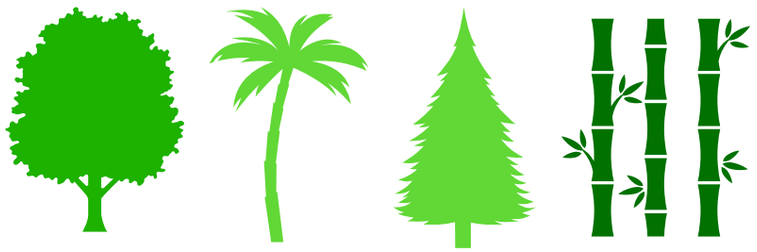
```
public class Plant{
    private String name;

    public Plant(String name){
        this.name = name;
     }
    public String getName(){
        return name;
    }
    public String toString(){
        return getName();
    }

    public boolean equals(Object obj){
        if (obj instanceof Plant){
            Plant p = (Plant)obj;
            return p.getName().equals(this.getName());
        }
        return false;
    }
}
```

Example 2

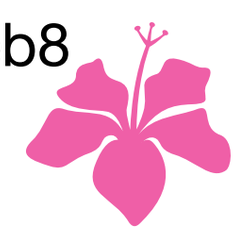
```
Collection<Plant> plants;
//plants is initialised and filled with Plant objects
Plant hp = new Plant(“Hibiscus”);
if(plants.contains(hp)){
    plants.remove(hp);
}
```

//Plant@56a081b3

Outcome:
Poui
Coconut
Pine
Bamboo



Plant@36d3b4a2
Plant@26df8cb7
Plant@47b4f4f9
Plant@fbe288c
Plant@c401eb8

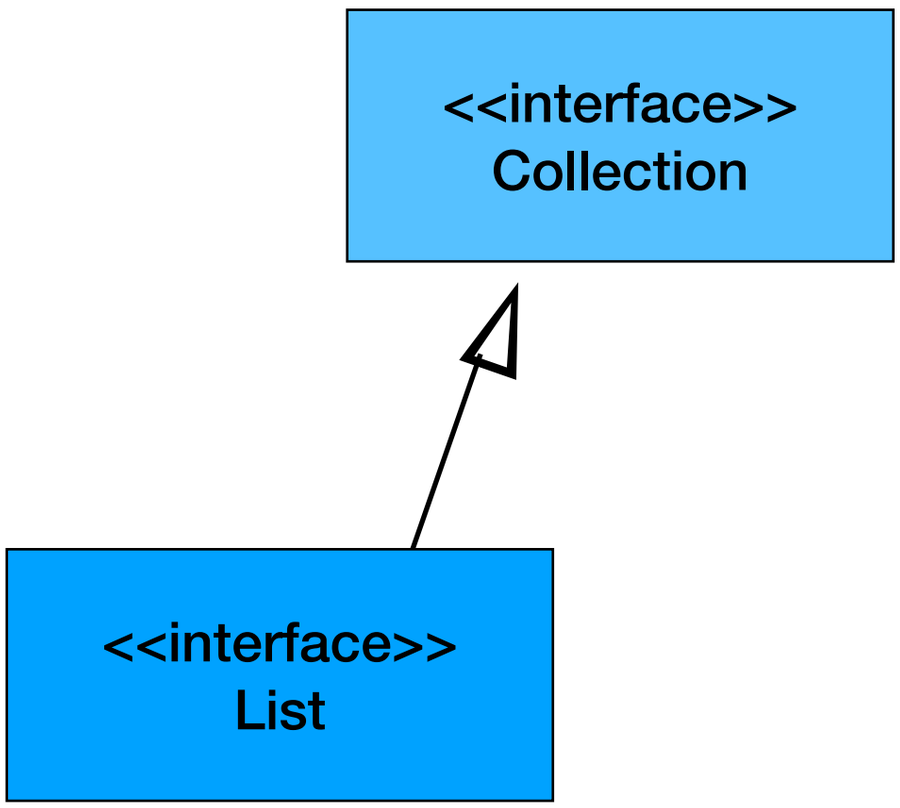


Example 3

```
Collection<Plant> plants;// = new TreeSet<>();
//plants is initialised and filled with Plant objects
PineTree pt = new PineTree(“Conifer”);
plants.add(pt);
```

Outcome:

Interfaces in the Java Collections Framework



The List Interface

The List interface represents an ordered collection of objects (also known as a sequence). Lists allow precise control over where each element is inserted.

Each element in a List has an index, or position in the list, and elements can be inserted, queried, and removed by index.

The first element of a List has an index of zero.

The last element in a List has an index of size( )-1.

Lists typically allow duplicate elements.

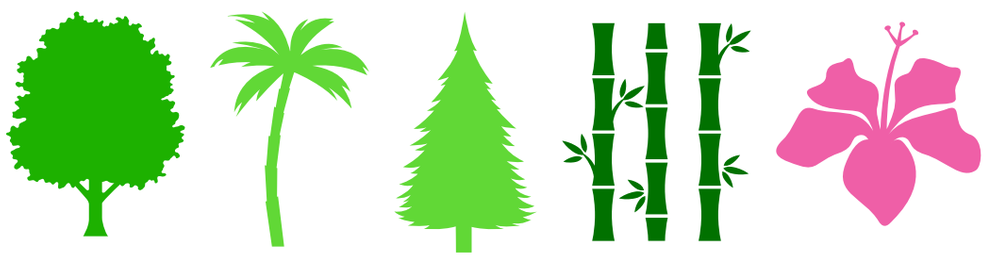
The List Interface

Method	Description
void add (int index, E element)	Inserts the object supplied as an argument in position index of the List. The other elements in the List are shifted down one position
E get (int index)	Returns the element at position index of the List
int indexOf(Object o)	Returns the position of the object supplied as an argument int he List, or -1 if it does not find a match. indexOf( ) uses the equals( ) method of the contained objects to check for equality with the object supplied as an argument.
E remove(int index)	Removes the object at position index of the List
E set(int index, E element)	Inserts the object supplied as element in position index of the List, overwriting the element that was there previously, if any. It returns the element that was previously stored at that position; otherwise it returns null

Example 4

```
Collection<Plant> plants = new LinkedList<>();
//plants is initialised, filled with 5 Plant objects
int numPlants = plants.size();   //returns 5
Plant p = plants.get(0); //fails

if( plants instanceof LinkedList){
    LinkedList<Plant> plantList = (LinkedList<>)plants;
    Plant p = plantList.get(0);
    System.out.println(p)
}
```

plants collection:



Outcome:

Example 5

```
Collection<Plant> plants = new LinkedList<>();
//plants is initialised, filled with 5 Plant objects
int numPlants = plants.size();   //returns 5
Plant p = plants.remove(0);
  // compilation error- why?
```

plants collection: