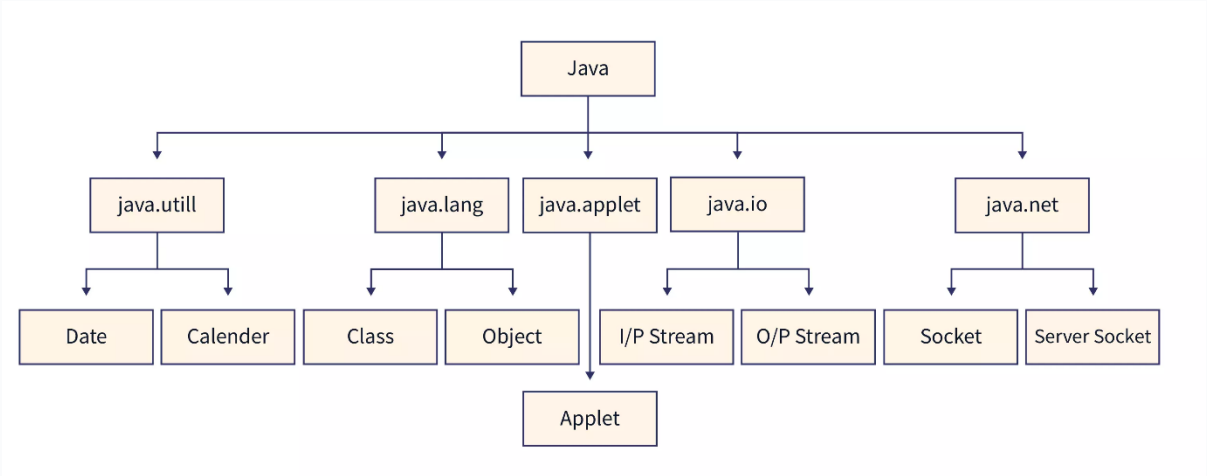
java.util

Dr. Monika Patel
Asst Professor
Computer Dept
Durga College Raipur(C.G.)

Java.util package contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes. This reference will take you through simple and practical methods available in java.util package.

To support the needs of a software developer, Java provides various built-in and pre-written functionalities in the form of packages. These built-in packages contain various kinds of classes and interfaces that can be used to develop better and maintainable code. Some of the built-in Java packages are shown in the figure given below:



One of the most important Java packages is the Utility (or Util) package. The Util package in Java contains several pre-written classes that can act as a base to solve commonly occurring problems in software development. It provides basic and essential source code snippets to the programmers that can lead to clean and maintainable code.

What Does import java.util Do in Java?

To load the basic utility classes and interfaces provided by the java.util package, we need to use Java's **import** keyword. The **import** keyword is used to access and load the package and its respective classes into the Java program. The syntax to import the package or its classes is given below:

```
import package_name.class_name;
import package_name.*;
```

Here, the first import statement is used to load a certain class from the specified package, whereas the second statement is used to import the whole package into the Java program. For the Util package in Java:

```
import java.util.*;
```

The above statement is used to load all the basic necessary functionalities provided by the Util package in Java.

java.util Package Import Statement Example

Let's look at some examples to understand the use of **import** statements while loading the Util package in Java:

Example 1: To Load the Whole Util Package

```
import java.util.*;
```

As discussed, this statement will load all the pre-written classes and interfaces provided by the Java Utility package.

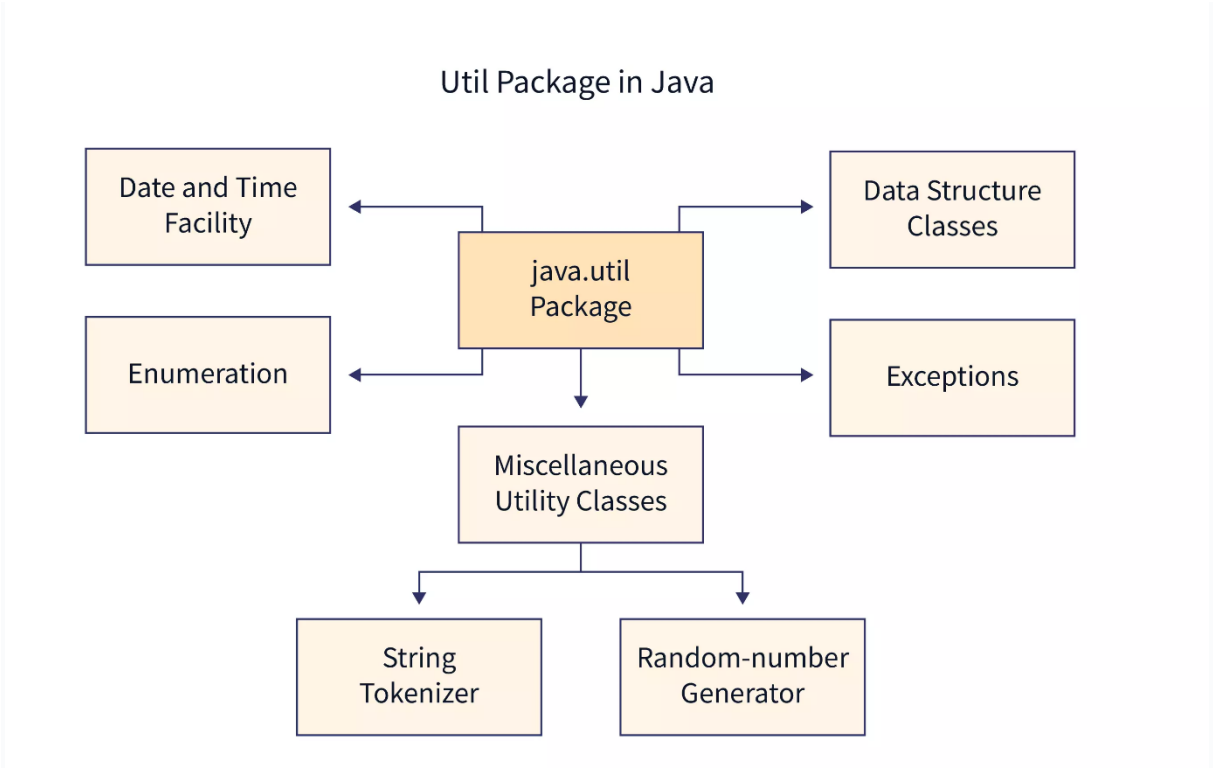
Example 2: To Load a Specific Class from the Util Package

```
import java.util.Scanner;
```

The above statement is importing the **Scanner** class available in the **java.util** package. This class is widely used to take input from the user in a Java program.

What is the Use of java.util Package in Java?

The Util package in Java consists of several components that provide basic necessities to the programmer. These components include:



- **Data Structure Classes:** The **java.util** package contains several pre-written data structures like Dictionary, Stack, LinkedList, etc. that can be used directly in the program using the **import** statements.
- **Date and Time Facility:** The **java.util** package provides several date and time-related classes that can be used to create and manipulate date and time values in a program.
- **Enumeration:** It provides a special interface known as Enumeration that can be used to iterate through a set of values.
- **Exceptions:** It provides classes to handle commonly occurring exceptions in a Java program.
- **Miscellaneous Utility Classes:** The **java.util** package contains essential utilities such as the string tokenizer and random-number generator.

What is Collections Framework in Java?

Collections Framework in Java is a special part of the **java.util** package that can be used to represent and manipulate collections. It provides easy storage and organization of a group of objects (or collection). It includes pre-written classes, interfaces, and algorithms under a unified architecture.

Conclusion

- Util package in Java is a built-in package that contains several pre-written utility classes and interfaces.
- The **import java.util.*;** statement can be used to load the contents of the Util package in a Java program.
- It consists of components such as data structures, exceptions, enumerations, utility classes, etc.
- It allows us to write clean and maintainable code.
- It contains the Collections Framework which is used to represent and manipulate collections.

Java ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package.

The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want. The syntax is also slightly different:

Example

Create an ArrayList object called **cars** that will store strings:

```
importjava.util.ArrayList;// import the ArrayList class



ArrayList<String> cars =newArrayList<String>();// Create an ArrayList object
```

- We can not create an array list of the primitive types, such as int, float, char, etc. It is required to use the required wrapper class in such cases. For example:

1. ArrayList<**int**> al = ArrayList<**int**>(); // does not work
2. ArrayList<Integer> al = **new** ArrayList<Integer>(); // works fine

Method	Description
void add (int index, E element)	It is used to insert the specified element at the specified position in a list.
boolean add (E e)	It is used to append the specified element at the end of a list.
boolean addAll (Collection<? extends E> c)	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
boolean addAll (int index, Collection<? extends E> c)	It is used to append all the elements in the specified collection, starting at the specified position of the list.
void clear ()	It is used to remove all of the elements from this list.
void ensureCapacity(int requiredCapacity)	It is used to enhance the capacity of an ArrayList instance.
E get(int index)	It is used to fetch the element from the particular position of the list.
boolean isEmpty()	It returns true if the list is empty, otherwise false.

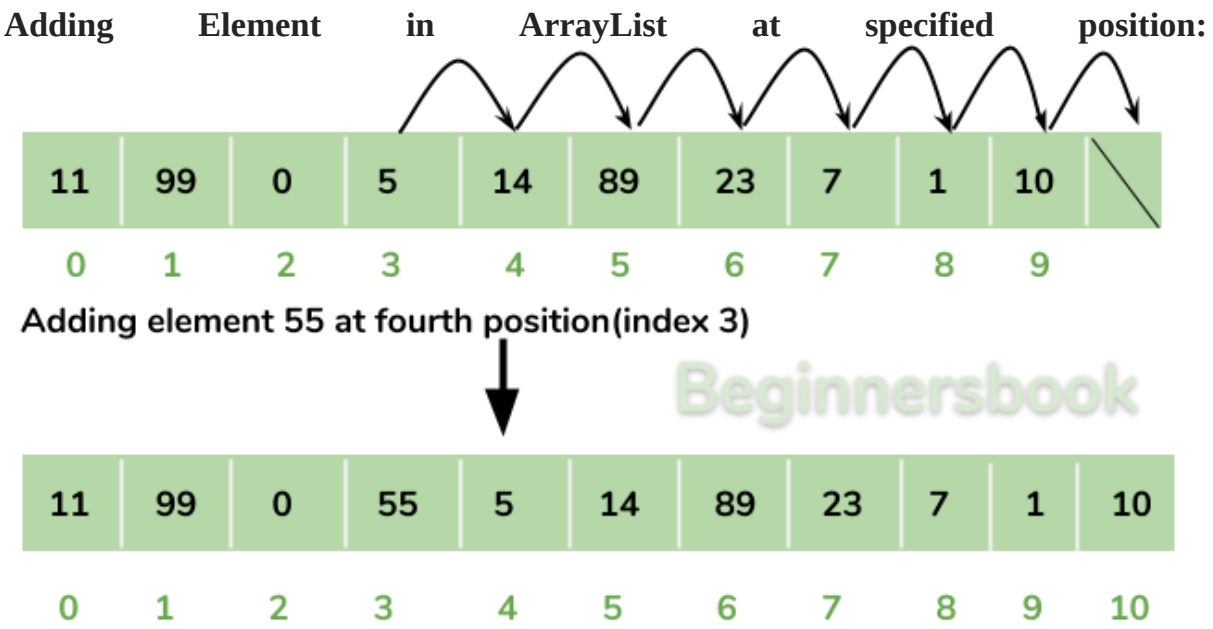
```
import java.util.ArrayList;

classMain{
publicstaticvoidmain(String[] args){
// create ArrayList
    ArrayList<String> languages = new ArrayList<>();

// add() method without the index parameter
    languages.add("Java");

    languages.add("C");
    languages.add("Python");
    System.out.println("ArrayList: " + languages);
  }
}
```

Adding elements to Arraylist in java



You can add elements to an ArrayList by using **add() method**. This method has couple of variations, which you can use based on the requirement.

For example: If you want to add the element at the end of the List then you can simply call the add() method like this:

```
arrList.add("Steve");//This will add "Steve" at the end of List
```

Change an element in ArrayList

You can use the **set method** to change an element in ArrayList. You need to provide the **index** and **new element**, this method then updates the element present at the **given index** with the **new given element**.

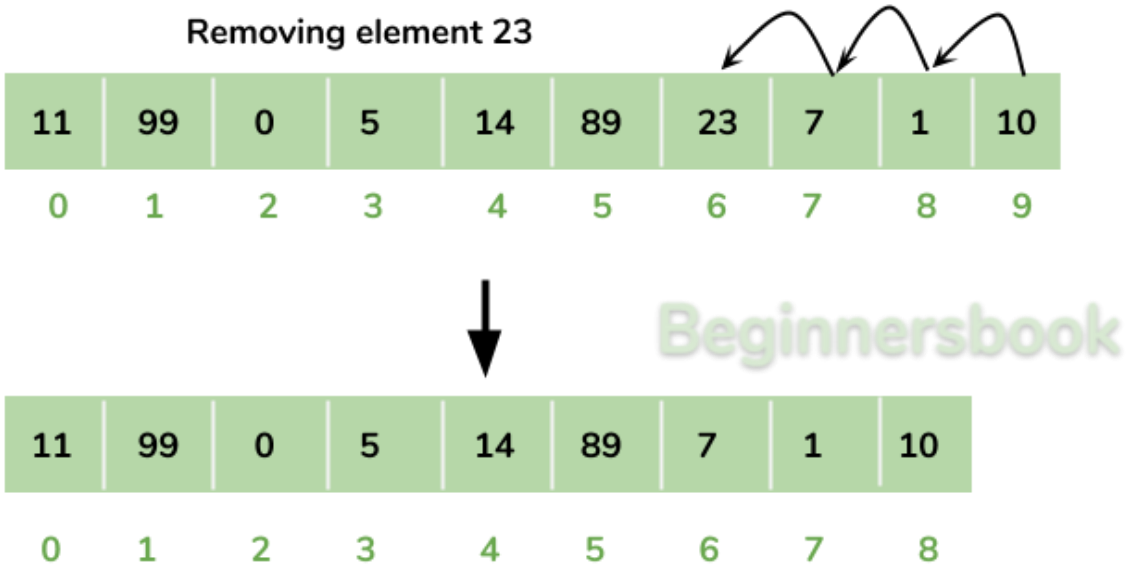
In the following example, we have given the index as 0 and new element as "Lucy" in the set() method. The method updated the element present at the index 0 ("Jim") with the new String element "Lucy".

```
import java.util.ArrayList;
publicclassJavaExample{
publicstaticvoid main(String[] args){
ArrayList<String> names =newArrayList<String>();
    names.add("Jim");
    names.add("Jack");
    names.add("Ajeet");
    names.add("Chaitanya");
    names.set(0,"Lucy");
System.out.println(names);
}
```

```
}
```

How to remove element from Arraylist in Java?

Removing Element from ArrayList:



You can use **remove() method** to remove elements from an ArrayList. Similar to add() method, this method also has couple of variations.

For example:

```
import java.util.*;
classJavaExample{
publicstaticvoid main(String args[]){
ArrayList<String> alist=newArrayList<String>();
    alist.add("Steve");
    alist.add("Tim");
    alist.add("Lucy");
    alist.add("Pat");
    alist.add("Angela");
    alist.add("Tom");
//displaying elements
System.out.println(alist);

//Removing "Steve" and "Angela"
    alist.remove("Steve");
    alist.remove("Angela");

//displaying elements
System.out.println(alist);

//Removing 3rd element
    alist.remove(2);

//displaying elements
System.out.println(alist);
}
}
```