Handling Date and Time in Java:
=========================

java.util.Date
java.util.Calendar

from java 1.8

Demo.java:

```
package com.masai;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.ZonedDateTime;

public class Demo {

	public static void main(String[] args)   {

		LocalDate ld= LocalDate.now();

		System.out.println(ld);

		LocalTime lt= LocalTime.now();

		System.out.println(lt);

		LocalDateTime ldt = LocalDateTime.now();

		System.out.println(ldt);

		ZonedDateTime zdt= ZonedDateTime.now();

		System.out.println(zdt);


	}

}
```

G - Era(AD BC)

y - year(yy(18) or yyyy(2018))

M - Month(M(9) or MM(09) or MMM(Sep))(MMMMM--September)

d - day(d(23) or dd(23) or ddd(023))

E - day in a weak(E (sun))(EEEE--Sunday)

a - am pm

h - hour in am or pm (1-12)

hh - hour in am or pm (01-12)

H - hour of day in 24 hour form (0-23)

HH - hour of day in 24 hour form (00-23)

m - minute (4)

mm - minute (04)

s - second (4)

ss - second(04)

Collection framework:
===================

Collection : if we want to represent a group of object as a single unit (single object) then we should collection.

framework: the main objective of a f/w is to ease developer work.

--it is semi-implemented architecture.
--A f/w comprises some abstract design with some built-in behaviour(functionality) in order to use it, we need to insert our functionality in various places of f/w

---A s/w f/w is a universal,reusable s/w platform to develop a s/w application,products, or solutions.

Date structure:
============

---array,
---stack
---Queue
---LinkedList
---BinarySearch (hashing)
---sorting algorithm

Collection f/w:
==============

--it defines several classes and interfaces which can be used to represent/arrange group of objectss as a single unit/object.

***we can group multiple objects as a single object by using arrays.

--as we know that in java arrays are also treated as an object.

```
A[] arr = new A[3]; //one object is created i.e array obj.and 3 A class variable is created with the value null.

arr[0] = new A();
arr[1] = new A();
```

arr[2] = new A();

limitaiton of array:
===============

1. size is fixed, we can not increase or decrease it dynamically.

2.it supports homogenious type of elements.

--this limitation we can overcome by taking Object class array.

Object[] or= new Object[3];

or[0] = new A();
or[1] = new Student();
or[2] = new Employee();

3. array concept is not implemented based on readymade method support. for each activity even for printing the elements from an array we need to write the logic manually.

--to overcome the above limitation we need to use collection f/w.

--Collections are growable and shrinkable in natute.
--collection can hold both homogenious and heteregenious elements.
--every collection classes r implemented based on some standard data-structure, hence readymade method support is available for most of the requirement.

***All the collection f/w related classes and interfaces belongs to java.util package.

---Collection having 2 section:

1.normal collection (here we manage object uniformally/individually)

2.Map (here we manage objects in key-value pair)

List : ---> when we need to arrange the elements in sequence(index based manner) and duplicate elements are allowed

Set :- when we need uniqueness (duplicate elements are not allowed)

Queue: when we need to arrange the element for prior to processing,.(FIFO is bydefault but we can manipulate)

Collection interface:
================

--it is the foundation upon which the collection f/w is built.

--it declares some of the core methods that all collection classes will have.

methods of Collection interface:

int size(); //how many elements are there in that collection object.

boolean isEmpty();

boolean contains(java.lang.Object); // searching an element

java.util.Iterator<E> iterator(); // this method inherited from Iterable interface

java.lang.Object[] toArray(); //to convert any collection object elements to normal Object[] array

<T> T[] toArray(T[]); // to convert any collection array to Object[] array

boolean add(Object obj); // to add any element(Object) to any collection classes

boolean remove(Object obj);

boolean containsAll(Collection col);

boolean addAll(Collection col);

boolean removeAll(Collection col);

boolean retainAll(Collection col);

public abstract void clear(); // clear out all the elements from the collection

  public abstract boolean equals(java.lang.Object); these methods are overriden from Object clas
  public abstract int hashCode();

//these methos add in java 1.8 v
  public java.util.Spliterator<E> spliterator();
  public boolean removeIf(java.util.function.Predicate<? super E>);
  public java.util.stream.Stream<E> stream();
  public java.util.stream.Stream<E> parallelStream();

Note: there is no any concrete class which implements Collection interface directly.

Iterable: this interface introduced in java 1.5 and from java 1.5 onwards Collection interface extends this interface. it belogs to java.lang package.

--this interface has only one abstract method:

public abstract Iterator iterator();

--the return type of this iterator() method is Iterator(I) interface,
--this Iterator interface belongs to java.util package.

--in addition to the one abstract method, this Iterator interface has 2 default method as well:

  public void forEach(java.util.function.Consumer<? super T>);
  public java.util.Spliterator<T> spliterator();

List(I):
=====

--it is the child interface of the Collection interface and declares the behaviour of a collection to preserve the sequence of an element.

--elements can be inserted and accessed by their position using zero based index.

--here insertion order will be preserved and duplicates elements are allowed.

--in addition to the Collection interface methods ,List interface defines some of its own methods also they are:

public Object get(int index);

Object remove(int index);

Object set(int index, Object obj); //assign the obj in specified index and return the overritten object

*****Note: Collection f/w only supports Objects, primitives are not allowed.

add(Object obj)

ArrayList(c):
=========

--it is the implementation of List interface.

--it dynamically increase and decrease in size.

--ArrayList class is the best choice if our frequent operation is retrieval based on index.

--duplicates are allowed.

--null insertion is possible (multiple null values)

```
ArrayList al = new ArrayList();
```

```
ArrayList al = new ArrayList();

System.out.println(al);// []
```

--in the above statement we have create an empty AL object with the default initial capacity 10.

--once AL reaches to its max capacity then a new AL object will be created in the memory automatically with the capacity according the following formula:

newCapacity = (currentCapacity * 3/2) + 1

ArrayList al = new ArrayList(1000); // here new AL obj will be created with the initial capacity 1000.