

LIST COLLECTION:

```
import java.util.List;
```

```
import java.util.Iterator;
```

```
import java.util.LinkedList;
```

```
class Course {
```

```
    String courseName;
```

```
    public Course(String courseName) {
```

```
        this.courseName = courseName;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return courseName;
```

```
    }
```

```
}
```

```
class ListInterface {
```

```
    public static void main(String[] args) {
```

```
        List<Course> courseList = new LinkedList<>();
```

```
        courseList.add(new Course("Java"));
```

```
        courseList.add(new Course("Hibernate"));
```

```
        courseList.add(new Course("AngularJS"));
```

```
        // Accessing the list of courses Using Iterator
```

```
        Iterator<Course> courseIterator = courseList.iterator();
```

```
        System.out.println("Using Iterator to access the list of courses");
```

```
        while (courseIterator.hasNext()) {
```

```
            Course c = courseIterator.next();
```

```

        System.out.println(c); // toString() method has been overridden in the
Course class
    }

    // Accessing the list of courses Using for loop
    System.out.println("Using for loop to access the list of courses");
    for (int index = 0; index < courseList.size(); index++) {
        System.out.println(courseList.get(index));
    }

    // Accessing the list of courses Using enhanced for loop (for-each)
    System.out.println("Using enhanced for loop to access the list of courses");
    for (Course c : courseList) { // Can be read as: for each Course c in courseList
        System.out.println(c);
    }
}
}

```

OUTPUT:

```

Using Iterator to access the list of courses
Java
Hibernate
AngularJS
Using for loop to access the list of courses
Java
Hibernate
AngularJS
Using enhanced for loop to access the list of courses
Java
Hibernate
AngularJS

```

SET COLLECTION:

```

import java.util.LinkedHashSet;

import java.util.Set;

```

```

class SetInterface{

    public static void main(String[] args) {

        // Creating a new Set object of type Integer

        Set<Integer> numberSet = new LinkedHashSet<>();


        // Adding elements to the set
        numberSet.add(12);
        numberSet.add(24);
        numberSet.add(12);


        // Displaying the Set
        System.out.println(numberSet);

    }

}

```

OUTPUT:

```
[12, 24]
```

MAP COLLECTION:

```

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

```

```

class MapInterface {

    public static void main(String[] args) {

        Set<Course> courseSet1 = new HashSet<>();

        courseSet1.add(new Course("Java"));
    }

}

```

```
courseSet1.add(new Course("DBMS"));
```

```
Set<Course> courseSet2 = new HashSet<>();
```

```
courseSet2.add(new Course("PHP"));
```

```
courseSet2.add(new Course("HTML"));
```

```
courseSet2.add(new Course("CSS"));
```

```
Map<Integer, Set<Course>> studentCourses = new HashMap<>();
```

```
studentCourses.put(1001, courseSet1);
```

```
studentCourses.put(1002, courseSet2);
```

```
// Retrieving the set of Courses by studentID using get() method
```

```
Set<Course> courseSet = studentCourses.get(1001);
```

```
System.out.println("Retrieving the set of Courses by studentID: ");
```

```
System.out.println(courseSet);
```

```
// Iterating over the set of keys using for-each loop
```

```
Set<Integer> setOfKeys = studentCourses.keySet();
```

```
System.out.println("Iterating over the set of keys using for-each loop: ");
```

```
for (Integer i : setOfKeys) {
```

```
    System.out.println(studentCourses.get(i));
```

```
}
```

```
// Iterating over the collection using values() method
```

```
System.out.println("Iterating over the collection using values() method: ");
```

```
for (Set<Course> courses : studentCourses.values()) {
```

```
    System.out.println(courses);
```

```
}
```

```
}
```

```
}
```

```

class Course {
    String courseName;

    public Course(String courseName) {
        this.courseName = courseName;
    }

    @Override
    public String toString() {
        return courseName;
    }
}

```

OUTPUT:

```

Retrieving the set of Courses by studentID:
[Java, DBMS]
Iterating over the set of keys using for-each loop:
[Java, DBMS]
[HTML, PHP, CSS]
Iterating over the collection using values() method:
[Java, DBMS]
[HTML, PHP, CSS]

```

DATE API

```

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

class Demonstrator
{
    public static void main (String[] args) {
        System.out.println("*****LocalDate*****");
    }
}

```

```

    LocalDate registrationDate = LocalDate.now(); //Creates an object with system date

    System.out.println("Today's date(System date): "+registrationDate);

    LocalDate lastDate = registrationDate.plusDays(3); // Adding 3 days to the registration date

    System.out.println("Adding 3 days: "+lastDate);

    if(LocalDate.now().isBefore(lastDate)) {          // Checking whether current date is before the
last date

        System.out.println("Please take the assessment as soon as possible");

    }

    System.out.println();

    System.out.println("*****LocalTime*****");

    LocalTime startTime = LocalTime.now(); //Creates an object with system time

    System.out.println("Start Time: "+startTime); //The time being returned here is according to the
region/locale/zone in which the application is hosted.


    LocalTime endTime = startTime.plusHours(1); //Adding 1 hour

    System.out.println("End Time: "+endTime); //The time being returned here is according to the
region/locale/zone in which the application is hosted.


    LocalTime current = LocalTime.now();

    int hour = current.getHour();          // Getting the hours, minutes and seconds components

    int minute = current.getMinute();

    int second = current.getSecond();

    System.out.println("Hour: "+hour+" Minute: "+minute+" Second: "+second); //The time being
returned here is according to the region/locale/zone in which the application is hosted.


    System.out.println();

    System.out.println("*****LocalDateTime*****");

    LocalDateTime dateTime = LocalDateTime.now();

    System.out.println("Date and Time: "+dateTime); //The time being returned here is according to
the region/locale/zone in which the application is hosted.

    System.out.println(dateTime.minusDays(3)); // Subtracting 3 days and output time is in ETC

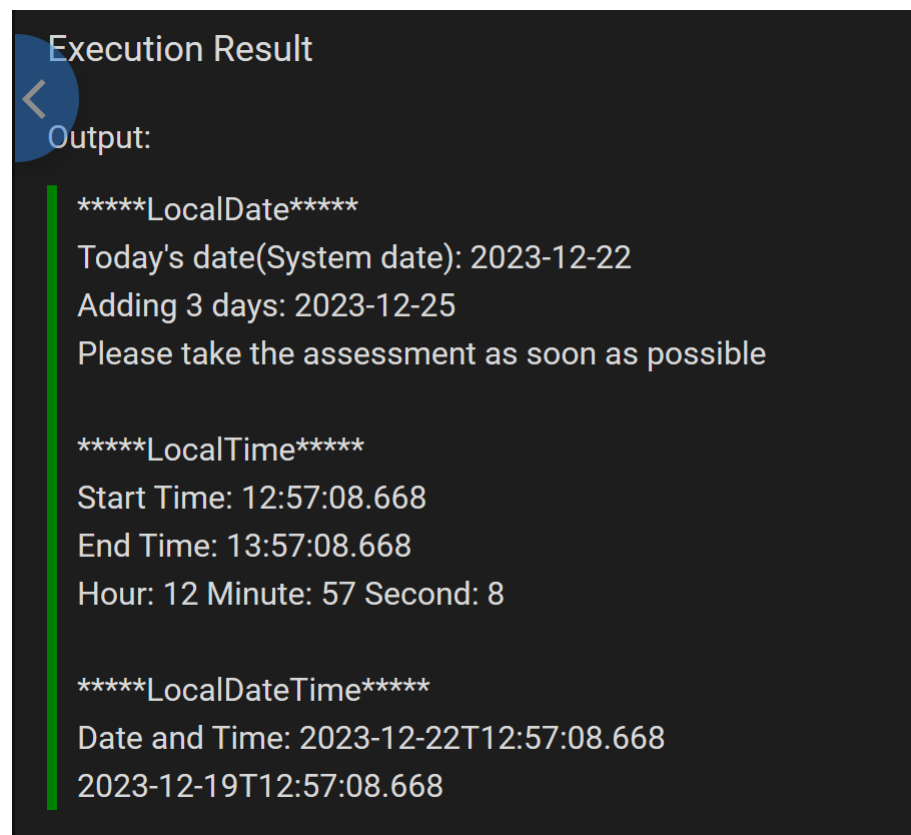
    //Run the above program in your system in order to see the difference

}

```

```
}
```

OUTPUT:



```
Execution Result
<
Output:
*****LocalDate*****
Today's date(System date): 2023-12-22
Adding 3 days: 2023-12-25
Please take the assessment as soon as possible

*****LocalTime*****
Start Time: 12:57:08.668
End Time: 13:57:08.668
Hour: 12 Minute: 57 Second: 8

*****LocalDateTime*****
Date and Time: 2023-12-22T12:57:08.668
2023-12-19T12:57:08.668
```

```
import java.time.ZonedDateTime;
import java.time.ZoneId;
import java.time.LocalDate;
import java.time.Period;
import java.time.temporal.ChronoUnit;
import java.time.format.DateTimeFormatter;

class Demonstrator
{
    public static void main (String[] args) {
        //ZonedDateTime is used to change the time according to zones
        System.out.println("*****ZonedDateTime*****");
        System.out.println("ZonedDateTime Now: "+ZonedDateTime.now());
    }
}
```

```
System.out.println("ZonedDateTime Europe:
"+ZonedDateTime.now(ZoneId.of("Europe/Athens")));
```

```
//Period
```

```
System.out.println();
```

```
System.out.println("*****Period*****");
```

```
LocalDate present = LocalDate.now();
```

```
LocalDate later = present.plusDays(3);
```

```
Period period = Period.between(present, later);
```

```
int lops = period.getDays();
```

```
System.out.println("Difference in days: "+lops);
```

```
//ChronoUnit
```

```
System.out.println();
```

```
System.out.println("*****ChronoUnit*****");
```

```
LocalDate today = LocalDate.now();
```

```
System.out.println("Today's date: "+today);
```

```
LocalDate after = today.plus(1,ChronoUnit.MONTHS);
```

```
System.out.println("Date after 1 month: "+after);
```

```
//DateTimeFormatter
```

```
System.out.println();
```

```
System.out.println("*****DateTimeFormatter*****");
```

```
LocalDate paySlipDate = LocalDate.now();
```

```
System.out.println(paySlipDate); // Output: Current Date
```

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MMM/yyyy");
```

```
System.out.println(paySlipDate.format(formatter)); // Output: Current Date
```

```
}
```

```
}
```

OUTPUT:

Output:

```
*****ZonedDateTime*****  
ZonedDateTime Now: 2023-12-22T12:58:08.898+05:30[Asia/Kolkata]  
ZonedDateTime Europe: 2023-12-22T09:28:08.900+02:00[Europe/Athens]  
  
*****Period*****  
Difference in days: 3  
  
*****ChronoUnit*****  
Today's date: 2023-12-22  
Date after 1 month: 2024-01-22  
  
*****DateTimeFormatter*****  
2023-12-22  
22/Dec/2023
```

```
import java.util.Calendar;
```

```
class CalendarDemo{
```

```
    public static void main(String[] args){
```

```
        Calendar cal=Calendar.getInstance();//Create Calendar
```

```
        System.out.println("Current Date is\t:" + cal.getTime() + "\n");//print current date
```

```
        cal.set(1992,0,1);//set the date to 1st Jan,1992
```

```
        System.out.println("Date after 1st modification\t:" + cal.getTime() + "\n");
```

```
        cal.add(Calendar.MONTH,3);//add 3 months
```

```
        System.out.println("Date after adding 3 months\t:" + cal.getTime() + "\n");
```

```
        cal.add(Calendar.YEAR,-3);//subtract 3 years
```

```
        System.out.println("Date after subtracting 3 years\t:" + cal.getTime() + "\n");
```

```
    }
```

```
}
```

OUTPUT:

Execution Result

Output:

Current Date is :Fri Dec 22 12:59:11 IST 2023

Date after 1st modification :Wed Jan 01 12:59:11 IST 1992

Date after adding 3 months :Wed Apr 01 12:59:11 IST 1992

Date after subtracting 3 years :Sat Apr 01 12:59:11 IST 1989