ASSIGNMENT- (APIs & ANNOTATION)

Q. **Program to Display current Date and Time in java.**

Ans-
```java
import java.time.*;

public class time {
    public static void main(String[] args) {
        LocalDate date = LocalDate.now();
        System.out.println(date);

        int day = date.getDayOfMonth();
        int month = date.getMonthValue();
        int year = date.getYear();
        System.out.println(day + "/" + month + "/" + year);

        LocalTime time = LocalTime.now();
        System.out.println(time);
        int hour = time.getHour();
        int min = time.getMinute();
        int sec = time.getSecond();
        System.out.println(hour + ":" + min + ":" + sec);
    }
}
```

Q2. **Write a Program to convert a date to a string in the format “MM/DD/YY”.**

Ans-
```java
import java.time.*;

public class time {
    public static void main(String[] args) {
        LocalDate date = LocalDate.now();
        System.out.println(date);

        int day = date.getDayOfMonth();
        int month = date.getMonthValue();
        int year = date.getYear();
        System.out.println(day + "/" + month + "/" + year);

        LocalTime time = LocalTime.now();
        System.out.println(time);
        int hour = time.getHour();
        int min = time.getMinute();
        int sec = time.getSecond();
        System.out.println(hour + ":" + min + ":" + sec);
    }
}
```

Q3. **What is the difference between collections and streams? Explain with Example.**

Ans- Streams:-

- It doesn't store data, it operates on the source data structure i.e collection.
- They use functional interfaces like lambda which makes it a good fit for programming languages.
- Java Streams are consumable i.e; to traverse the stream , it needs to be created every time.
- Java streams support both sequential and parallel processing.
- Streams are iterated internally by just mentoring the operations.

```
Example: import java.util.Arrays;

import java.util.*;
import java.util.stream.*;

public class Stream_API {
    public static void main(String[] args) {

        List<Integer> list = Arrays.asList(2, 4, 5, 8, 9);

        Stream<Integer> streamData = list.stream();

        streamData.forEach(n -> System.out.println(n));

        // streamData.forEach(n -> System.out.println(n * 2));
    }
}
```

Collections:-

- It stores/ hold all the data that the data structure currently has in a particular data structure like Set , List or map.
- They don't use Functional Interfaces.
- They are non-consumable i.e; can be traversable multiple times without creating it again.
- It supports parallel processing can be very helpful in achieving high performance.
- Collections are iterated externally using loops.

```
Example:  import java.util.*;


public class collection_Class {
    public static void main(String[] args) {

        ArrayList al = new ArrayList<>();
        al.add(100);
        al.add(50);
        al.add(150);
        al.add(25);
        al.add(75);
        al.add(125);

        System.out.println(al);
        Collections.sort(al);
        System.out.println(al);
```

Q4. **What is enums in java? Explain with an example.**

Ans- We can use enum to define a group of named constants. Enums are used to represent a collection of related constants that have a common purpose. Each constant in an enum is an instance of the enum type, and they are typically defined as public static final fields. Here's an example of how to define an enum in Java:

```
 enum Week {

    MON, TUE, WED, THU, FRI, SAT, SUN;
}

public class Enum {

    enum Result {
        PASS, FAIL, NR;
    }

    public static void main(String[] args) {
        Week w = Week.MON;
        System.out.println(w);
        int index = Week.SUN.ordinal();
        System.out.println(index);

        Week[] wr = Week.values();
        for (Week w1 : wr) {
            System.out.println(w1 + ":" + w1.ordinal());
        }

        // Result r = Result.PASS;
        // System.out.println(r);
    }
}
```

Here we define an enum called "DayOfWeek" that represents the days of the week. The enum has seven constants, each representing a day of the week. The constants are defined in all uppercase letters by convention.

Q5. **What are in built annotations in java?**

Ans- built-in annotations in Java:

- @Override
- @Deprecated
- @SuppressWarnings
- @FunctionalInterface
- @Retention
- @Target
- @Documented
- @Inherited

These built-in annotations in Java are used to provide additional information to the Java compiler and other tools. They help improve code readability, maintainability, and safety by enforcing specific rules and behaviours in Java code