**APIs and Annotation solution**

1. Program to display current date and time in Java

java

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Main {

public static void main(String[] args) {

LocalDateTime currentDateTime = LocalDateTime.now();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

String formattedDateTime = currentDateTime.format(formatter);

System.out.println("Current Date and Time: " + formattedDateTime);

}

}

2. Program to convert a date to a string in the format "MM/dd/yyyy":

java

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

public class Main {

public static void main(String[] args) {

LocalDate date = LocalDate.now();

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

String formattedDate = date.format(formatter);

System.out.println("Formatted Date: " + formattedDate);

}

}

3. Difference between collections and streams with an example:

- Collections in Java represent groups of objects, and they provide various data structures like lists, sets, and maps to store and manipulate elements. They are mutable and can be modified over time.

- Streams, introduced in Java 8, are sequences of elements that support various aggregate operations like filtering, mapping, and reducing. Streams are typically used to perform bulk operations on collections or arrays. They are immutable and do not modify the underlying data source.

java

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Main {

public static void main(String[] args) {

List<String> names = Arrays.asList("John", "Alice", "Bob", "Mary");

// Example using collections (list)

List<String> filteredNames = names.stream()

.filter(name -> name.startsWith("A"))

.collect(Collectors.toList());

System.out.println("Filtered Names using Stream: " + filteredNames);

// Example using collections (list)

List<String> filteredNamesList = names.stream()

.filter(name -> name.startsWith("A"))

.collect(Collectors.toList());

System.out.println("Filtered Names using Collection: " + filteredNamesList);

}

}

4. Enums in Java:

- Enums in Java are special data types that allow a variable to be a set of predefined constants. They provide type safety and can be used to represent fixed sets of constants like days of the week, months, etc.

java

public enum Day {

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY

}

public class Main {

public static void main(String[] args) {

Day today = Day.MONDAY;

System.out.println("Today is: " + today);

}

}

5. Built-in annotations in Java:

- Java provides several built-in annotations such as @Override, @Deprecated, @SuppressWarnings, etc.

- These annotations are used to provide metadata about the code, control compiler warnings, and specify behavior.

- For example, @Override annotation is used to indicate that a method is overriding a superclass method.