Assignment - Java 8 Feature - 2

Q1)Implement following functional interfaces from java.util.function using lambdas: 1. Consumer 2. Supplier 3. Predicate 4. Function.

```java
package Java_8_2.Q1;
import java.util.function.*;

public class Q1 {



        public static void main(String[] args) {
            Consumer<String> consumer = String s -> System.out.println("Consumed: " + s);
            consumer.accept( t: "Java 8");

            System.out.println("\n\n");

            Supplier<String> supplier = () -> "Supplied value";
            System.out.println(supplier.get());

            System.out.println("\n\n");

            Predicate<Integer> predicate = Integer n -> n > 10;
            System.out.println("Is 15 > 10? " + predicate.test( t: 15));

            System.out.println("\n\n");

            Function<String, Integer> function = String::length;
            System.out.println("Length of 'Lambda': " + function.apply( t: "Lambda"));
        }
    }
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=44591 -Dfil
Consumed: Java 8



Supplied value



Is 15 > 10? true



Length of 'Lambda': 6

Process finished with exit code 0
```

Q2)Create and access default and static method of an interface.

```java
package Java_8_2.Q2;
interface Vehicle {  2 usages  1 implementation

    default void start() {  1 usage
        System.out.println("Starting the vehicle...");
        System.out.println("\n\n");
    }

    static void showType() {  1 usage
        System.out.println("Vehicles can be bikes, cars, trucks, etc.");
        System.out.println("\n\n");
    }
}
class Bike implements Vehicle {  2 usages
}
class InterfaceExample {  1 usage
    public static void Demo() {  1 usage
        Bike myBike = new Bike();
        myBike.start();
        Vehicle.showType();
    }
}
public class Q2 {
    public static void main(String[] args) {
        InterfaceExample.Demo();
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=39209 -Df
Starting the vehicle...



Vehicles can be bikes, cars, trucks, etc.



Process finished with exit code 0
```

Q3)Sum all the numbers greater than 5 in the integer list using streams

```java
package Java_8_2.Q3;
import java.util.Arrays;
import java.util.List;

public class Q3 {



        public static void main(String[] args) {
            List<Integer> numbers = Arrays.asList(1, 4, 6, 8, 3, 9);

            System.out.println("Number of elements in list: ");

            for (Integer number : numbers) {
                System.out.print(number+"\t");
            }

            System.out.println("\n\n");
            int sum = numbers.stream() Stream<Integer>
                    .filter( Integer n -> n > 5)
                    .mapToInt(Integer::intValue) IntStream
                    .sum();

            System.out.println("Sum of numbers > 5: " + sum);
        }
    }
```

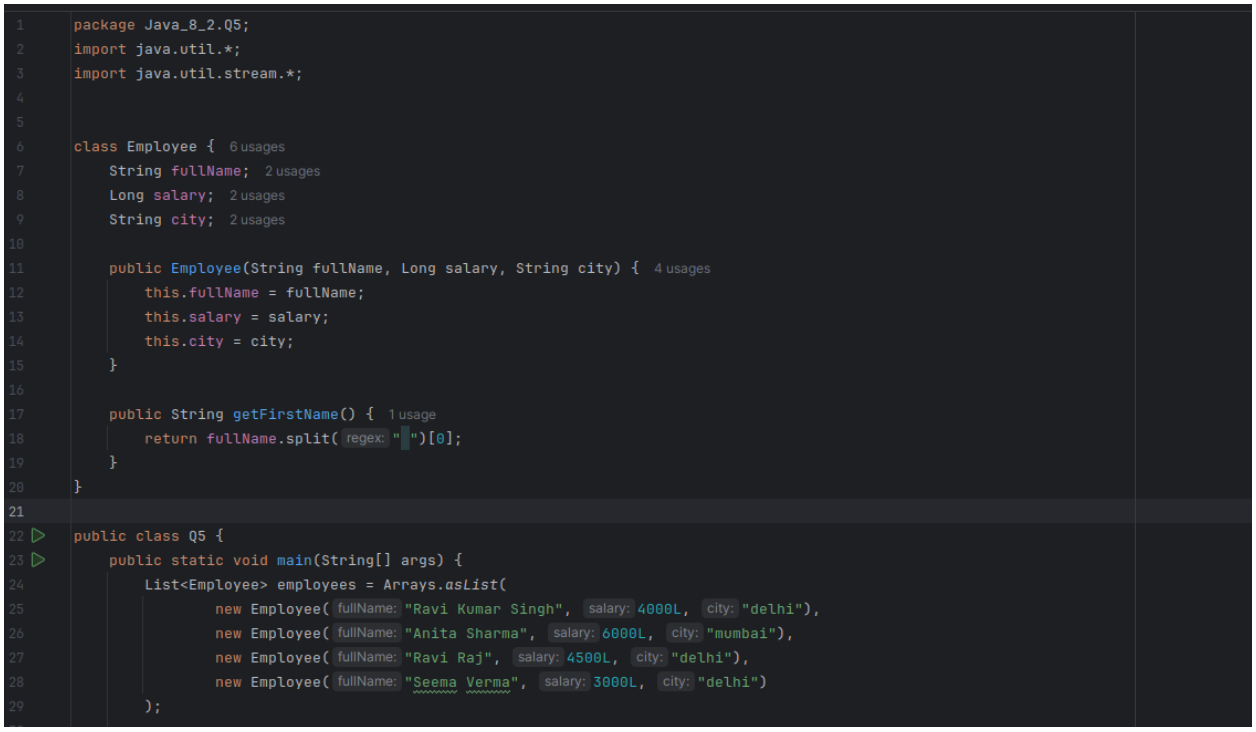
```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=36411 -Dfile.
Number of elements in list:
1   4   6   8   3   9


Sum of numbers > 5: 23

Process finished with exit code 0
```

Q4)Write a program to showcase the use of optional class

```java
package Java_8_2.Q4;

import java.util.Optional;

public class Q4 {
    public static void main(String[] args) {
        Optional<String> name = Optional.ofNullable( value: null);

        System.out.println("Is name present? " + name.isPresent());

        String value = name.orElse( other: "Default Name");
        System.out.println("Value: " + value);

        name.ifPresent( String n -> System.out.println("Uppercase: " + n.toUpperCase()));
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/
Is name present? false
Value: Default Name

Process finished with exit code 0
```

Q5)Given a list of objects of following class: class Employee{ String fullName; Long salary; String city; } Get list of all unique firstNames of employees where their salary is less than 5000 and who live in delhi. Note: Full name is concatenation of first name, middle name and last name with single space in between.



```java
package Java_8_2.Q5;
import java.util.*;
import java.util.stream.*;


class Employee {  6 usages
    String fullName;  2 usages
    Long salary;  2 usages
    String city;  2 usages

    public Employee(String fullName, Long salary, String city) {  4 usages
        this.fullName = fullName;
        this.salary = salary;
        this.city = city;
    }

    public String getFirstName() {  1 usage
        return fullName.split( regex: " ")[0];
    }
}

public class Q5 {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(
                new Employee( fullName: "Ravi Kumar Singh",  salary: 4000L,  city: "delhi"),
                new Employee( fullName: "Anita Sharma",  salary: 6000L,  city: "mumbai"),
                new Employee( fullName: "Ravi Raj",  salary: 4500L,  city: "delhi"),
                new Employee( fullName: "Seema Verma",  salary: 3000L,  city: "delhi")
        );
```

```java
    public String getFirstName() {  1 usage
        return fullName.split( regex: " ")[0];
    }
}

public class Q5 {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(
                new Employee( fullName: "Ravi Kumar Singh",  salary: 4000L,  city: "delhi"),
                new Employee( fullName: "Anita Sharma",  salary: 6000L,  city: "mumbai"),
                new Employee( fullName: "Ravi Raj",  salary: 4500L,  city: "delhi"),
                new Employee( fullName: "Seema Verma",  salary: 3000L,  city: "delhi")
        );

        List<String> uniqueFirstNames = employees.stream() Stream<Employee>
                .filter( Employee e -> e.salary < 5000 && "delhi".equalsIgnoreCase(e.city))
                .map(Employee::getFirstName) Stream<String>
                .distinct()
                .collect(Collectors.toList());

        System.out.println("Filtered First Names: " + uniqueFirstNames);
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=43397 -Dfile.encoding=
Filtered First Names: [Ravi, Seema]

Process finished with exit code 0
```

Q6)Using java 8 date/time api:

1. WAP to get two dates from user and print if the first date occurs before or after the second date supplied by the user.

```java
package Java_8_2.Q6;

import java.time.LocalDate;
import java.util.Scanner;


class MyDate {  4 usages
    private int year;  4 usages
    private int month;  4 usages
    private int day;  4 usages


    public MyDate(int year, int month, int day) {  2 usages
        this.year = year;
        this.month = month;
        this.day = day;
    }


    public int getYear() { return year; }  no usages
    public int getMonth() { return month; }  no usages
    public int getDay() { return day; }  no usages


    public LocalDate toLocalDate() {  2 usages
        return LocalDate.of(year, month, day);
    }


    @Override
    public String toString() {
        return year + "-" + String.format("%02d", month) + "-" + String.format("%02d", day);
    }
}

```

```java

public class Q6_1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter First Date:");
        System.out.print("Year: ");
        int year1 = sc.nextInt();
        System.out.print("Month: ");
        int month1 = sc.nextInt();
        System.out.print("Day: ");
        int day1 = sc.nextInt();

        MyDate firstDate = new MyDate(year1, month1, day1);
        System.out.println("Enter Second Date:");
        System.out.print("Year: ");
        int year2 = sc.nextInt();
        System.out.print("Month: ");
        int month2 = sc.nextInt();
        System.out.print("Day: ");
        int day2 = sc.nextInt();

        MyDate secondDate = new MyDate(year2, month2, day2);
        LocalDate d1 = firstDate.toLocalDate();
        LocalDate d2 = secondDate.toLocalDate();

        if (d1.isBefore(d2)) {
            System.out.println("First date is before second date.");
        } else if (d1.isAfter(d2)) {
            System.out.println("First date is after second date.");
        } else {
            System.out.println("Both dates are equal.");
        }
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=38041 -Dfile.encoding=
Enter First Date:
Year: 2025
Month: 06
Day: 23
Enter Second Date:
Year: 2025
Month: 06
Day: 24
First date is before second date.

Process finished with exit code 0
```

2. WAP to print current date and time in 3 different time zones.

```java
package Java_8_2.Q6;

import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

public class Q6_2 {
    public static void main(String[] args) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

        ZonedDateTime indiaTime = ZonedDateTime.now(ZoneId.of( zoneId: "Asia/Kolkata"));
        ZonedDateTime nyTime = ZonedDateTime.now(ZoneId.of( zoneId: "America/New_York"));
        ZonedDateTime tokyoTime = ZonedDateTime.now(ZoneId.of( zoneId: "Asia/Tokyo"));

        System.out.println("\nIndia Time: " + indiaTime.format(formatter));
        System.out.println("\nNew York Time: " + nyTime.format(formatter));
        System.out.println("\nTokyo Time: " + tokyoTime.format(formatter));
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=367

India Time: 2025-06-28 22:31:17

New York Time: 2025-06-28 13:01:17

Tokyo Time: 2025-06-29 02:01:17

Process finished with exit code 0
```