Java8 Features Assignment

Aditya kumar

//1

```java
import java.util.function.BiFunction;
interface Calculation {
    boolean isgreater(int a,int b);
}
interface Increment{
    int increment(int a);
}
interface ConcatSttring{
    String conString(String a,String b);
}
interface Toupper{
    String toupperstring(String a);
}
interface Add{
    Integer add(Integer a,Integer b);
}
 class lambdaPractice1 {
    public static void main(String[] args) {
        Calculation c = (a, b) -> { return  a > b? true :  false; };
        c.isgreater(5, 6);
        Increment i = (a) -> a += 1;
        System.out.println(i.increment(4));
        ConcatSttring concat = (a, b) -> a.concat(b);
        System.out.println(concat.conString("aditya", "kumar"));
        Toupper t = (a) -> a.toUpperCase();
        System.out.println(t.toupperstring("aditya"));
//Q2        Create a functional interface whose method takes 2 integers and return one integer.
        BiFunction<Integer,Integer,Integer> biFunction=(a,b)->a+b;
        System.out.println(biFunction.apply(5,9));
        Add addno=(a,b)->a+b;
        System.out.println(addno.add(4,6));
    }
}
```

3//

```java
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.function.BiFunction;
class MethodRreferenceExample {
    static int addno(int a, int b) {
        return a + b;
    }
    static int subno(int a, int b) {
        return a - b;
    }
}
class MethodReference3 {
    public static void main(String[] args) {
        //method reference using functional interface
        BiFunction<Integer, Integer, Integer> bi = MethodRreferenceExample::addno;
        System.out.println(bi.apply(4, 7));
        BiFunction<Integer, Integer, Integer> bi1 = MethodRreferenceExample::subno;
        System.out.println(bi.apply(4, 7));
        //method reference using instance::instanceMethod
        List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
        MyComparator myComparator = new MyComparator();
        Collections.sort(list, myComparator::compare);
        list.forEach(e -> System.out.println(e));
        Collections.sort(list, (a, b) -> myComparator.compare(a, b));
        //method reference using className::instanceMethod
        final List<Person> people = Arrays.asList(new Person("aditya"), new
Person("Rahul"));
        people.forEach(Person::printName);
        people.forEach(person -> person.printName());
        for (final Person person : people) {
            person.printName();
        }
    }
}
class MyComparator {
    public int compare(final Integer a, final Integer b) {
        return a.compareTo(b);
    }
}
class Person {
    private String name;
    public Person(final String name) {
        this.name = name;
    }
    public void printName() {
        System.out.println(name);
    }
}
```

//4

```
interface EmployeeInterface
{
    Employee getEmployee(String name,Integer age,String city);
}
class Employee {
    String name;
    Integer age;
    String city;
    public Employee(String name, Integer age, String city)
    {
        this.name=name;
        this.age=age;
        this.city=city;
        System.out.println("Name:"+this.name+" Age:"+this.age+" City:"+this.city);
    }
}
class EmployeeConstructorReference4
{
    public static void main(String[] args) {
        EmployeeInterface employeeInterface=Employee::new;
        employeeInterface.getEmployee("aditya",23,"pondicherry");
    }
}
```

//5
```
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;
 class LambdaOperation5{
    public static void main(String[] args) {
        Consumer<String> con=s-> System.out.println(s);
        con.accept("Aditya");
        Supplier<String> sup=()-> new String("hello world");
        System.out.println(sup.get());
        Predicate<Integer> pre=(s)->{ return s>5?true:false; };
        System.out.println(pre.test(10));
        Function<Integer,String> func=(a)-> String.valueOf(a);
        System.out.println(func.apply(0));
    }
}
```

//6

```
interface Featured{
    default void print(){
        System.out.println("default method of featured interface");
    }
    static void display(){
        System.out.println("static method of featured interface");
    }
}
 class Featured1 implements Featured{
  /*  public void print(){
        System.out.println("subclass featured1 overridden method");
    }*/
 }
public class DefaultAndStatic6 {
    public static void main(String[] arg){
        Featured.display();
        Featured obj1=new Featured1();
        obj1.print();
    }
}
```

//7

```
interface Featured2{
    default void print(){
        System.out.println("default method of featured interface");
    }
    static void display(){
        System.out.println("static method of featured interface");
    }
}
class Featured3 implements Featured{
   public void print(){
        System.out.println("subclass featured3 overridden method");
    }
}
public class OverrideDefaultMethod7 {
    public static void main(String[] arg){
```

```
		Featured.display();
		Featured obj1=new Featured3();
		obj1.print();
	}
}
```

//8

```
interface Featured4{
	default void display(){
		System.out.println("default method of featured4");
	}
}
interface Featured5{
	default void display1(){
		System.out.println("default method of Featured5");
	}
}
public class MultipleInheritance8 implements Featured4,Featured5{
	public static void main(String[] arg){
		Featured4 f4=new MultipleInheritance8();
		((MultipleInheritance8) f4).display1();
		f4.display();
	}
}
```

//9,10,11,12

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Q9Collect all the even numbers from an integer list.
//Q10Sum all the numbers greater than 5 in the integer list.
//Q11Find average of the number inside integer list after doubling it.
//Q12Find the first even number in the integer list which is greater than 3.
public class ListOperation9 {
	public static void main(String[] args) {
		List<Integer> list=
Arrays.asList(2,7,12,78,34,2,35,3,4,5,7,56,7,32,3,45,45,4,67,76,5,63);
		list.stream().filter(e->e%2==0).collect(Collectors.toList()).forEach(e->
System.out.println(e));
		System.out.println("Sum::"+list.stream().filter(e->e>5).mapToInt(i-
>i.intValue()).sum());
		System.out.println("Average after double::"+list.stream().mapToInt(e-
>e*2).average().getAsDouble());
		System.out.println("First even number greater than 3::"+list.stream().filter(e-
>e%2==0).filter(e->e>3).findFirst().get());
	}
}
```