Java8 Exercise

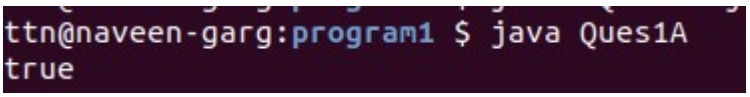
1. Write the following a functional interface and implement it using lambda:

(A) First number is greater than second number or not. Parameter (int ,int) Return boolean

Sol – Program Files Folder name – program2/Ques1A.java

```
interface Greater {
	boolean isGreaterThan(int a,int b);
}


public class Ques1A{
	public static void main(String[] args) {
		int a=5,b=4;
		Greater g= (x,y)-> x>y;
		System.out.println(g.isGreaterThan(a,b));
	}
}
```



(B) Increment the number by 1 and return incremented value. Parameter (int) Return int

Sol – Program Files Folder name – program2/Ques1B.java

```
interface Increment {
	int calculate(int a);
}
public class Ques1B{
	public static void main(String[] args){
		int a=10;
		Increment g= (x)-> x+1;
		System.out.println(g.calculate(a));
```

```
    }
}
```

```
ttn@naveen-garg:program1 $ java Ques1B
11
```

(C) Concatination of 2 string. Parameter (String , String) Return (String)

Sol – Program Files Folder name – program2/Ques1C.java

```
interface Concatenate{
    String operate(String a,String b);
}
public class Ques1C{
    public static void main(String[] args){
        Concatenate g= (x, y)-> x.concat(y);
        System.out.println(g.operate("Naveen","Garg"));
    }
}
```

```
ttn@naveen-garg:program1 $ java Ques1C
HelloWorld
```

(D) Convert a string to uppercase and return . Parameter (String) Return (String)

Sol – Program Files Folder name – program2/Ques1D.java

```
interface UpperCase {
    String operate(String a);
}
public class Ques1D{
    public static void main(String[] args){
        UpperCase g= (x)-> x.toUpperCase();
        System.out.println(g.operate("naveengarg"));
    }
}
```

```
ttn@naveen-garg:program1 $ java Ques1D
NAVEENGARG
```

2.Create a functional interface whose method takes 2 integers and return one integer.

Sol – Program Files Folder name – program2

```
interface Addable{
    int add(int a,int b);
}


public class Ques2{
    public static void main(String[] args) {

        Addable ad1=(a,b)->(a+b);
        System.out.println(ad1.add(100,20));
    }
}
```

```
ttn@naveen-garg:program2 $ javac Ques2.java
ttn@naveen-garg:program2 $ java Ques2
120
```

3. Using (instance) Method reference create and apply add and subtract method and using (Static) Method reference create and apply multiplication method for the functional interface created.

Sol – Program Files Folder name – program3

```
import java.util.function.BiFunction;
class AddSub {
        public int add(int a, int b) {
                return a + b;
        }
```

```
	public int sub(int a, int b) {
		return a - b;
	}
}

class Multiplication{
   public static int multiply(int a, int b){
	return a*b;
   }
}
public class Ques3{

	public static void main(String[] args) {
		AddSub op = new AddSub();
		System.out.println("\nUsing Instance Method Reference");
		BiFunction<Integer, Integer, Integer> add2 = op::add;
		System.out.println("Addtion = " + add2.apply(40, 5));

		BiFunction<Integer, Integer, Integer> sub2 = op::sub;
		System.out.println("Subtraction = " + sub2.apply(10, 56));

		System.out.println("\nUsing Static Method Reference");
		BiFunction<Integer, Integer, Integer> product =
Multiplication::multiply;
		int pr = product.apply(4, 8);
		System.out.println("Product = "+pr);
	}
}
```

```
ttn@naveen-garg:program3 $ java Ques3

Using Method Reference
Addtion = 45
Subtraction = -46
ttn@naveen-garg:program3 $ vim Ques3.java
ttn@naveen-garg:program3 $ javac Ques3.java
ttn@naveen-garg:program3 $ java Ques3

Using Instance Method Reference
Addtion = 45
Subtraction = -46

Using Static Method Reference
Product = 32
```

4.Create an Employee Class with instance variables (String) name, (Integer)age, (String)city and get the instance of the Class using constructor reference

Sol – Program Files Folder name – program4

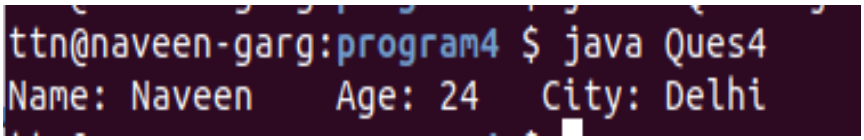
```
interface EmployeeSupplier{
    Employee get(String name,Integer age,String city);
}


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;
    }

    void printEmployee(){
        System.out.println("Name: "+name+"    Age: "+age+"   City: "+city);
    }
}
```

```
public class Ques4{
    public static void main(String[] args) {
        EmployeeSupplier supplier =Employee::new;
        Employee ob=supplier.get("Naveen",24,"Delhi");
        ob.printEmployee();
    }
}
```



5.Implement following functional interfaces from java.util.function using lambdas:

Sol – Program Files Folder name – program5

(1) Consumer

```
import java.util.*;
import java.util.function.Consumer;
class Product {
        private double price = 0.0;

        public void setPrice(double price) {
                this.price = price;
        }

        public void printPrice() {
                System.out.println(price);
        }
}

public class ConsumerQues5 {
        public static void main(String[] args) {
```

```
        Consumer<Product> updatePrice = p -> p.setPrice(9.9);
        Product p = new Product();
        updatePrice.accept(p);
        p.printPrice();
    }
}
```

```
ttn@naveen-garg:java8 $ javac ConsumerQues5.java
ttn@naveen-garg:java8 $ java ConsumerQues5
9.9
```

(2) Supplier

```
import java.util.*;
import java.util.function.Supplier;

class SupplierQues5{
    public static void main(String[] args) {
        int n = 10;
        display(() -> n + 35);
        display(() -> n + 110);
    }

    static void display(Supplier<Integer> arg) {
        System.out.println(arg.get());
    }
}
```

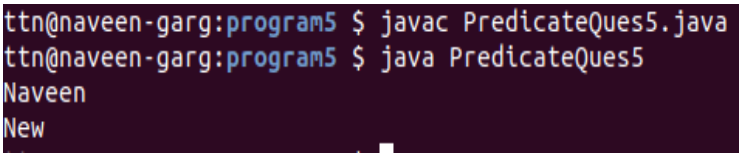
```
ttn@naveen-garg:program5 $ javac SupplierQues5.java
ttn@naveen-garg:program5 $ java SupplierQues5
45
120
```

(3) Predicate

```
import java.util.*;
import java.util.function.Predicate;
```

```
public class PredicateQues5{
	public static void main(String[] args) {
		List<String> names =
	   Arrays.asList("Naveen","New","Akash","Xy","ToTheNew");

	Predicate<String> p = (s)->s.startsWith("N");
	for (String st:names)
	{
	   if (p.test(st))
		System.out.println(st);
	}
	}
}
```



(4) Function

```
import java.util.*;
import java.util.function.Function;

class FuctionQues5 {
	public static void main(String[] args) {
		int n = 5;
		modifyValue(n, val-> val + 10);
		modifyValue(n, val-> val * 100);
	}

	public static void modifyValue(int v, Function<Integer, Integer> function){
		int result = function.apply(v);
```

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

```
ttn@naveen-garg:java8 $ javac FunctionQues5.java
ttn@naveen-garg:java8 $ java FuctionQues5
15
500
```

6. Create and access default and static method of an interface.

Sol – Program Files Folder name – program6

```
interface MyInterface{
    default void newMethod(){
        System.out.println("Default method");
    }

    static void anotherNewMethod(){
        System.out.println("Static method");
    }

    void Display(String str);
}
public class Ques6 implements MyInterface{
    public void Display(String str){
        System.out.println("String is: "+str);
    }

    public static void main(String[] args) {
        Ques6 obj = new Ques6();
        obj.newMethod();
        MyInterface.anotherNewMethod();
        obj.Display("Java 8");
```

```
    }
}
```

```
ttn@naveen-garg:program6 $ java Ques6
Default method
Static method
String is: Java 8
```

7. Override the default method of the interface.

Sol – Program Files Folder name – program7

Interfaces can have default methods with implementation in Java 8 on later.

Interfaces can have static methods as well, similar to static methods in classes.

//Code

```
interface MyInterface1
{
    default void show()
    {
        System.out.println("Default MyInterface1");
    }
}


interface MyInterface2
{
    // override show()
    default void show()
    {
        System.out.println("Default MyInterface2");
    }
}


class Ques7 implements MyInterface1, MyInterface2
{
```

```
    public void show()
    {
        MyInterface1.super.show();
        MyInterface2.super.show();
    }

    public static void main(String args[])
    {
        Ques7 d = new Ques7();
        d.show();
    }
}
```

```
ttn@naveen-garg:program7 $ javac Ques7.java
ttn@naveen-garg:program7 $ java Ques7
Default MyInterface1
Default MyInterface2
```

8.Implement multiple inheritance with default method inside interface.

Sol – Program Files Folder name – program8

```
interface ABC{
        default void abc(){
                System.out.println("default abc method");
        }

        default void print(){
                System.out.println("default print abc method");
        }
}

interface XYZ{
        default void xyz(){
```

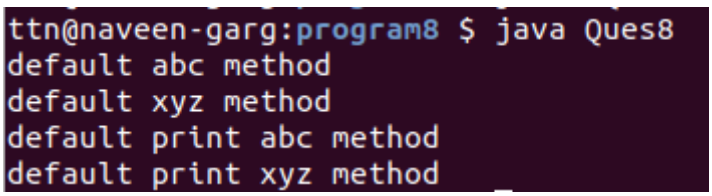
```
            System.out.println("default xyz method");
        }

        default void print(){
            System.out.println("default print xyz method");
        }
}

public class Ques8 implements ABC, XYZ {

        public void print(){
            ABC.super.print();
            XYZ.super.print();
        }

        public static void main(String[] args) {
            Ques8 obj = new Ques8();
            obj.abc();
            obj.xyz();
            obj.print();
        }
}
```



9.Collect all the even numbers from an integer list.

Sol – Program Files Folder name – program9

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

```
public class Ques9 {


  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    numbers.stream()
          .filter(value -> value % 2 == 0)
          .forEach(System.out::println);
  }
}
```

```
ttn@naveen-garg:program9 $ javac Ques9.java
ttn@naveen-garg:program9 $ java Ques9
2
4
6
8
10
```

10. Sum all the numbers greater than 5 in the integer list.

Sol – Program Files Folder name – program10

```
import java.util.Arrays;
import java.util.List;
import java.util.*;


public class Ques10{


  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    int total = numbers.stream().filter(value -> value > 5).mapToInt(value ->
value).sum();
    System.out.println("Sum all the numbers greater than 5: "+ total);
  }
}
```

```
ttn@naveen-garg:program10 $ vim Ques10.java
ttn@naveen-garg:program10 $ javac Ques10.java
ttn@naveen-garg:program10 $ java Ques10
Sum all the numbers greater than 5: 40
```

11. Find average of the number inside integer list after doubling it.

Sol – Program Files Folder name – program11

```
import java.util.Arrays;

import java.util.List;

import java.util.*;


public class Ques11{


  public static void main(String[] args) {

    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    double avg = numbers.stream().mapToDouble(value -> value +
value).average().orElse(-1);

    System.out.println("Average of the number inside integer list after doubling
it: "+ avg);

  }

}
```

```
ttn@naveen-garg:java8 $ java Ques11
Average of the number inside integer list after doubling it: 11.0
```

12. Find the first even number in the integer list which is greater than 3.

Sol – Program Files Folder name – program12

```
import java.util.Arrays;

import java.util.List;


public class Ques12{


  public static void main(String[] args) {

    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    int val = numbers.stream().filter(value -> value % 2 == 0).filter(value ->
value > 3).findFirst().get();
```

```
    System.out.println("First even number in the integer list which is greater
than 3 is "+val);
  }
}
```