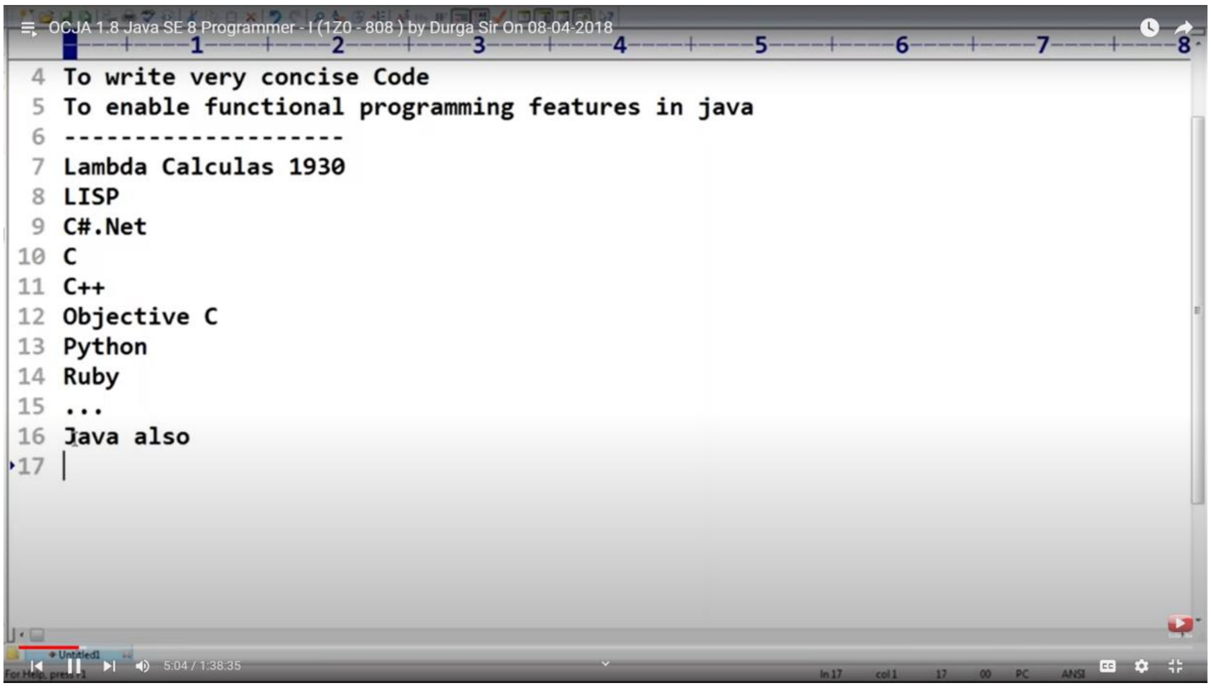
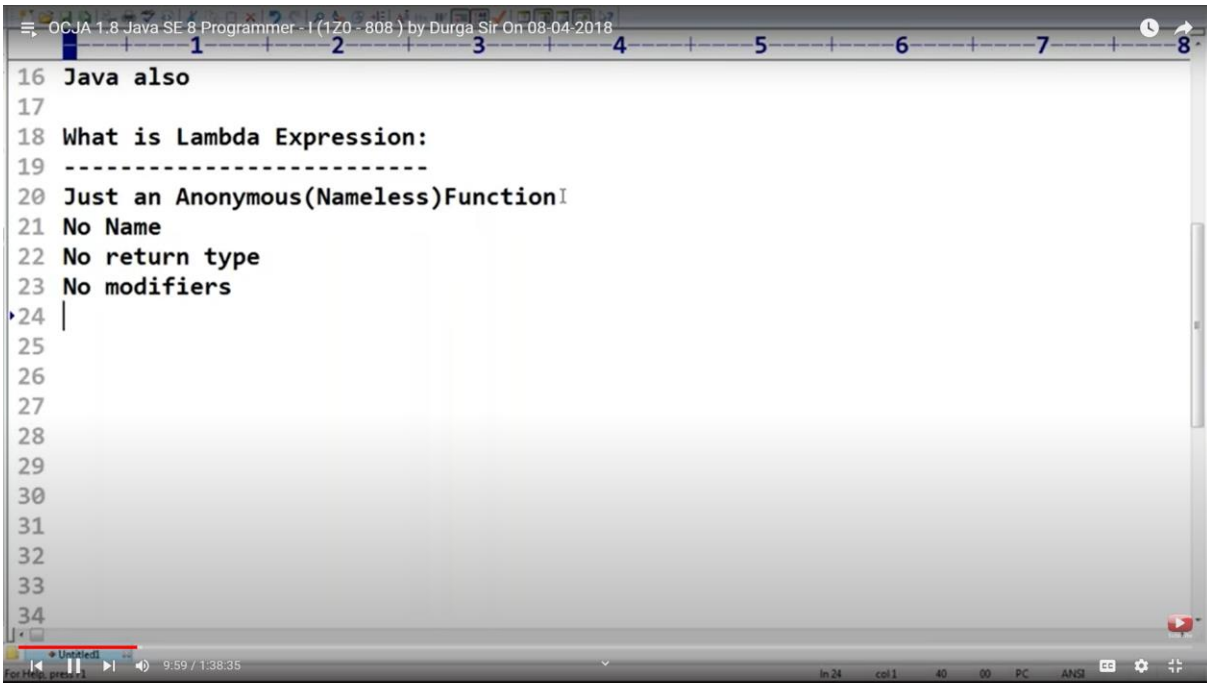

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
5:04 / 1:38:35

```
To write very concise Code
To enable functional programming features in java
--------------------
Lambda Calculas 1930
LISP
C#.Net
C
C++
Objective C
Python
Ruby
...
Java also
```


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
9:59 / 1:38:35

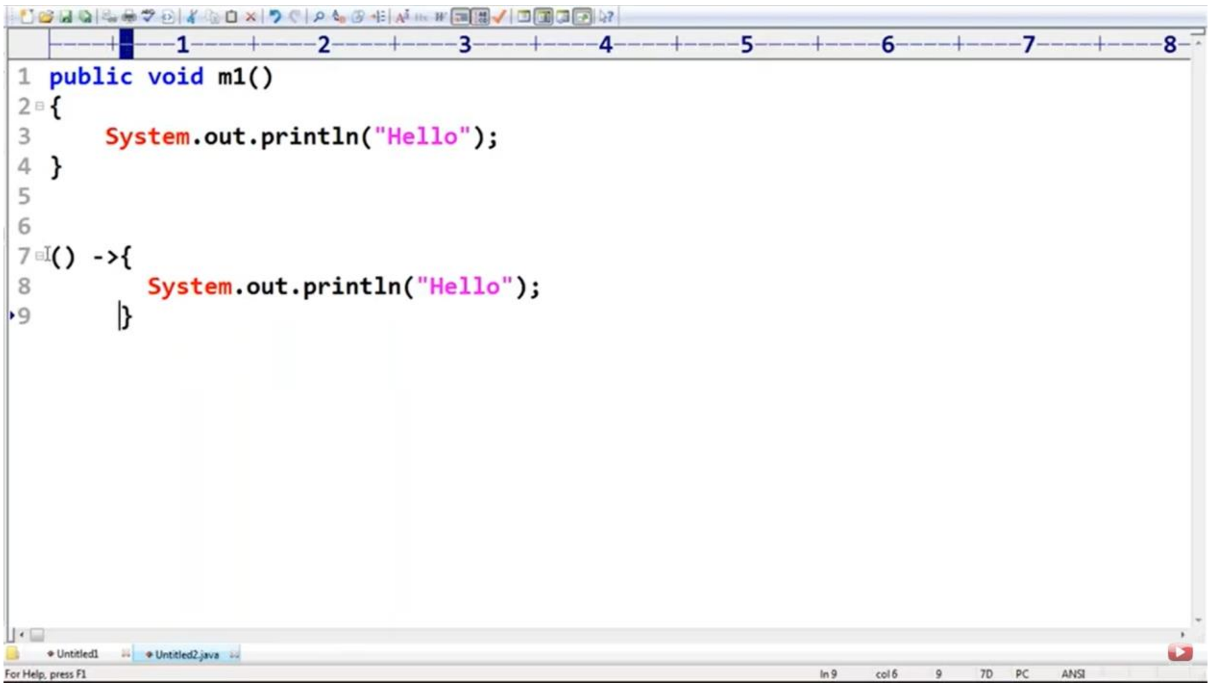
```
Java also

What is Lambda Expression:
-------------------------
Just an Anonymous(Nameless)Function
No Name
No return type
No modifiers
```

Remove name

Remove return type

Remove Modifier

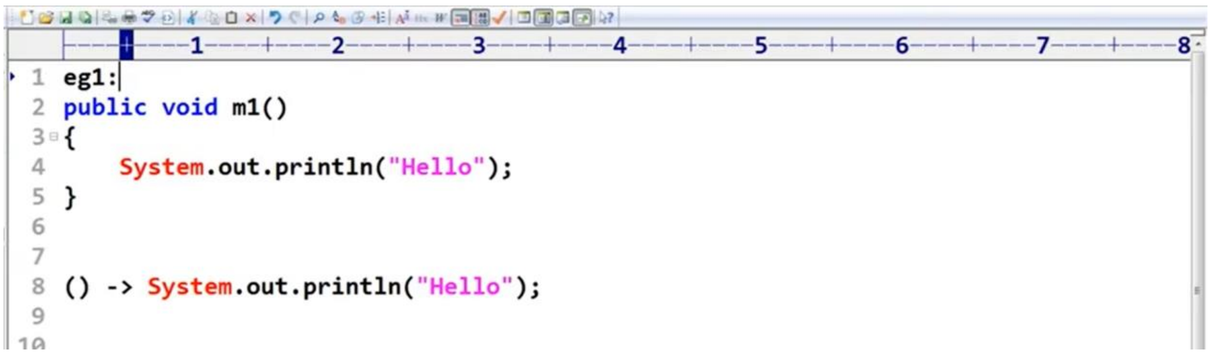
Just add an Arrow- this is anonymous function



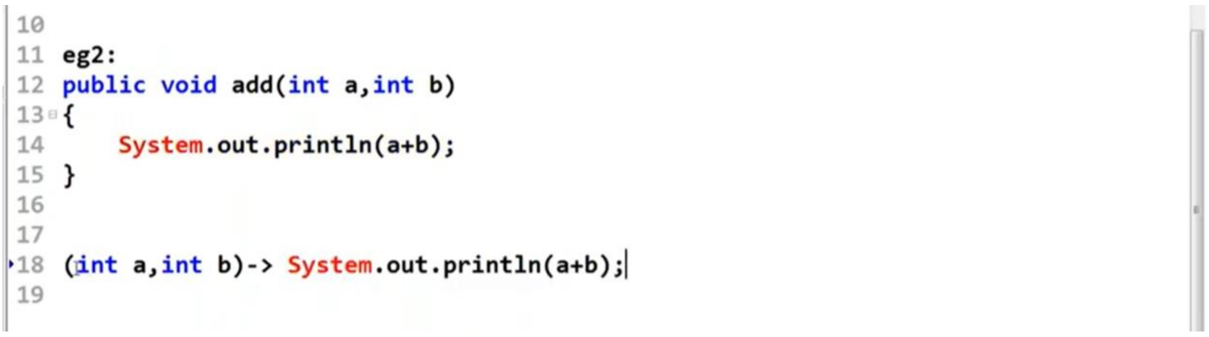
```
public void m1()
{
    System.out.println("Hello");
}


() ->{
        System.out.println("Hello");
      }
```

If its just 1 line u can remove curly braces also

()-> Sysout("Hello");

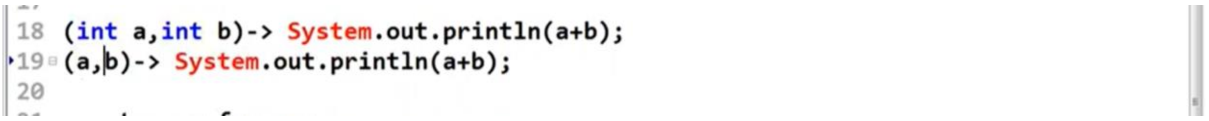


```
eg1:
public void m1()
{
    System.out.println("Hello");
}


() -> System.out.println("Hello");
```



```
eg2:
public void add(int a,int b)
{
    System.out.println(a+b);
}


(int a,int b)-> System.out.println(a+b);
```

Compiler can figure out the type by so we can remove it



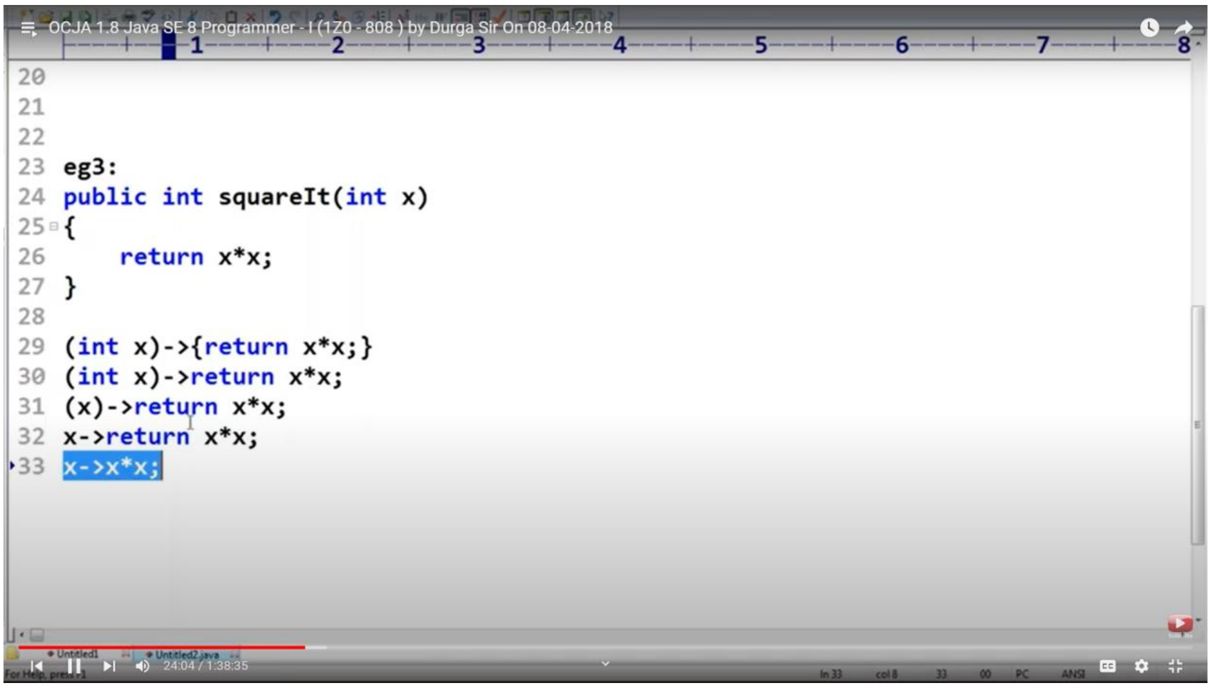
```
(int a,int b)-> System.out.println(a+b);
(a,b)-> System.out.println(a+b);
```

If only single line you can remove curly brackets

Since compile figures out the type you can remove the type from arg

Since we have only one argument u can also remove the parenthesis in args

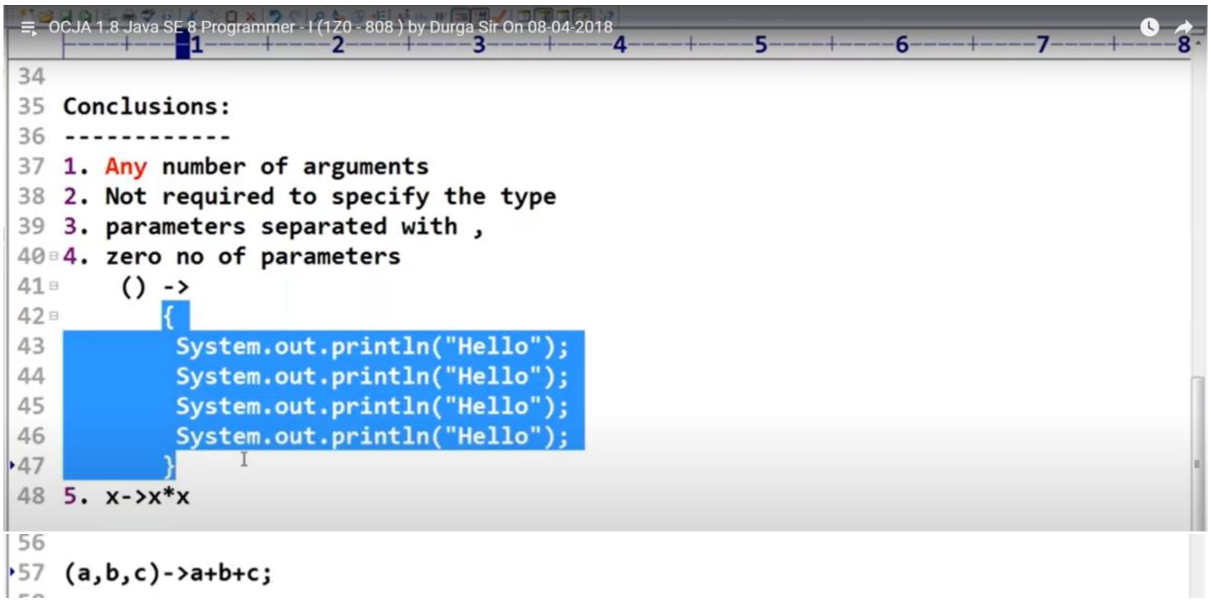
Return statement is also not required


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
24:04 / 1:38:35

```
eg3:
public int squareIt(int x)
{
    return x*x;
}

(int x)->{return x*x;}
(int x)->return x*x;
(x)->return x*x;
x->return x*x;
x->x*x;
```

If only one line then curly brackets are optional

Multi lines curly brackets are needed


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018

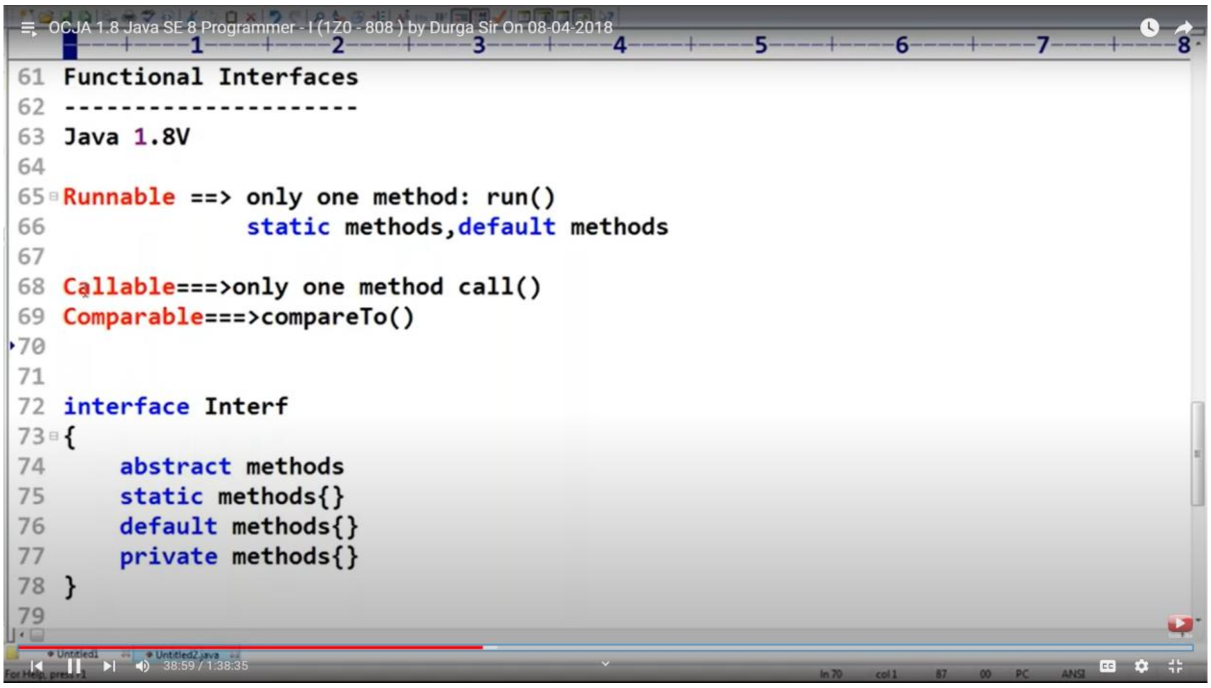
```
Conclusions:
------------
1. Any number of arguments
2. Not required to specify the type
3. parameters separated with ,
4. zero no of parameters
     () ->
         {
          System.out.println("Hello");
          System.out.println("Hello");
          System.out.println("Hello");
          System.out.println("Hello");
         }
5. x->x*x

(a,b,c)->a+b+c;
```

Lambda can be called using Functional Interface

If an interface contains only 1 abstract method its called Functional Interface.

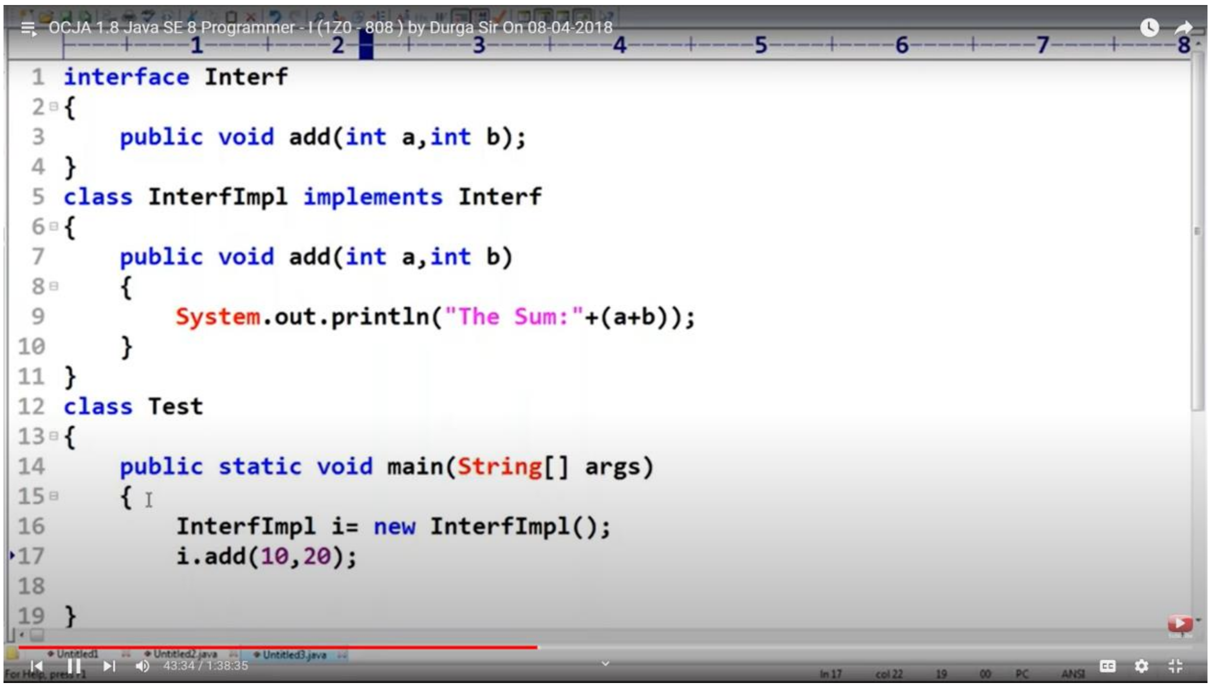
It may contain any no of Static methods, default and Private Methods does not matter. If it has only 1 abstract method its called Functional Interface

Runnable Interface has only 1 Abstract Method.


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
38:59 / 1:38:35

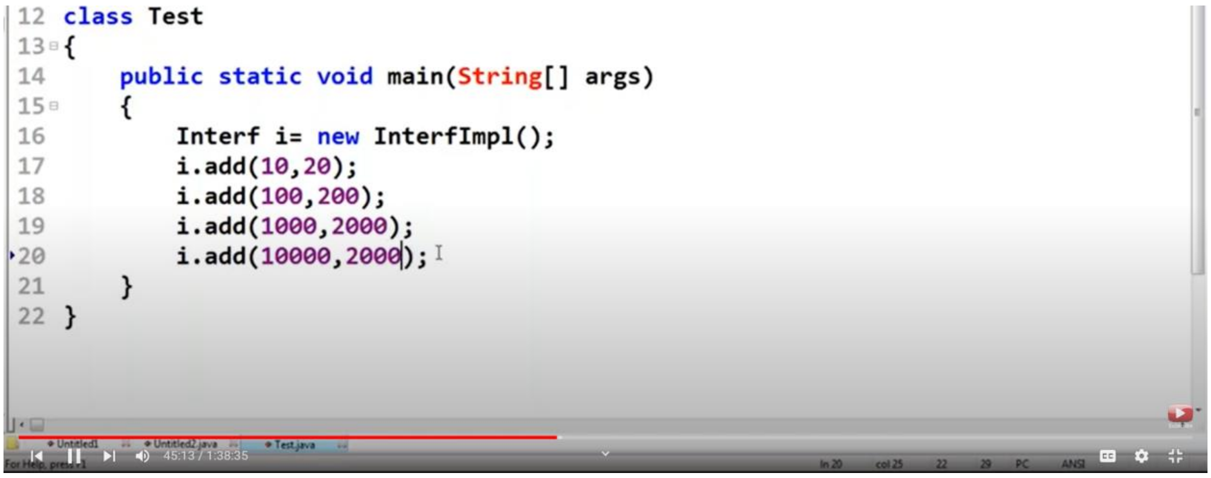
```
Functional Interfaces
---------------------
Java 1.8V

Runnable ==> only one method: run()
             static methods,default methods

Callable===>only one method call()
Comparable===>compareTo()


interface Interf
{
    abstract methods
    static methods{}
    default methods{}
    private methods{}
}
```

There is an interface and class implements it and we create an object and call it


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
43:34 / 1:38:35

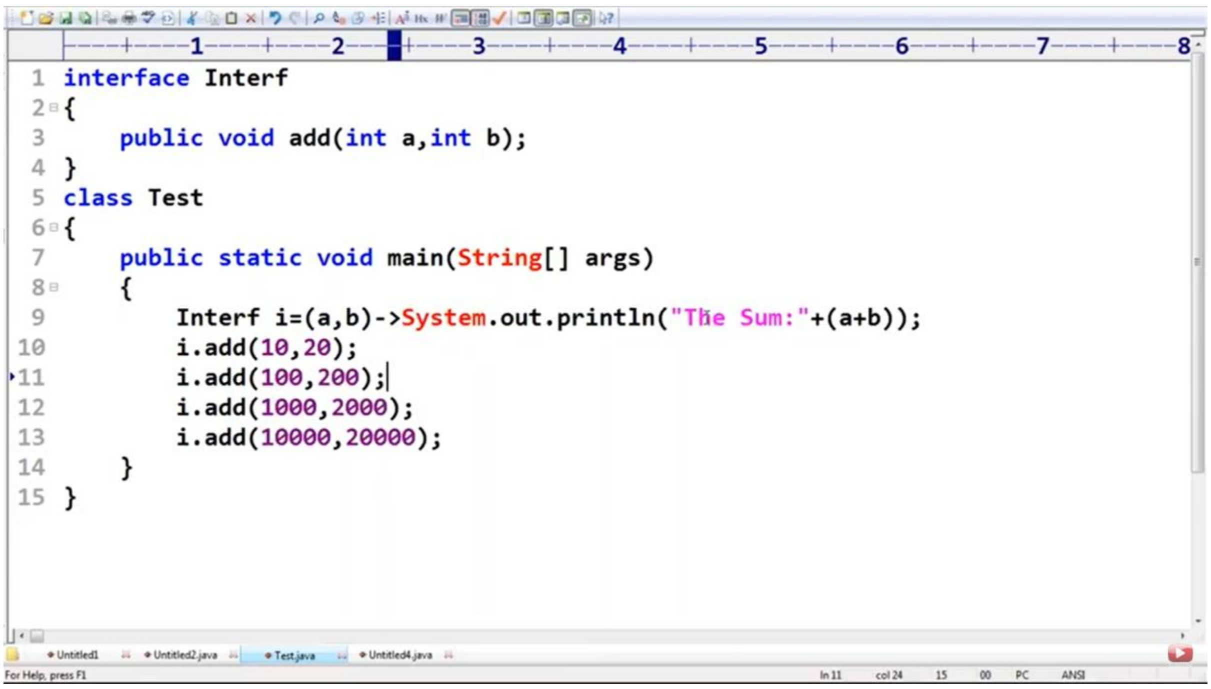
```java
interface Interf
{
    public void add(int a,int b);
}
class InterfImpl implements Interf
{
    public void add(int a,int b)
    {
        System.out.println("The Sum:"+(a+b));
    }
}
class Test
{
    public static void main(String[] args)
    {
        InterfImpl i= new InterfImpl();
        i.add(10,20);

}
```

Once an object is created u can call the method many times

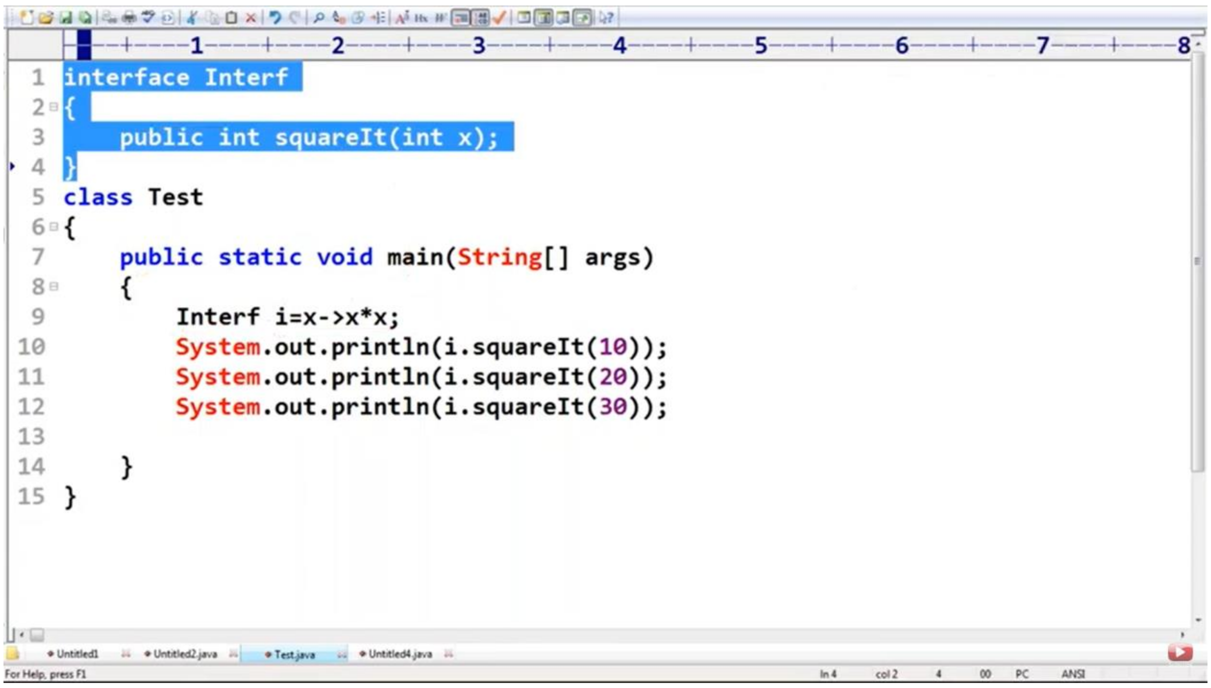


```java
class Test
{
    public static void main(String[] args)
    {
        Interf i= new InterfImpl();
        i.add(10,20);
        i.add(100,200);
        i.add(1000,2000);
        i.add(10000,2000);
    }
}
```

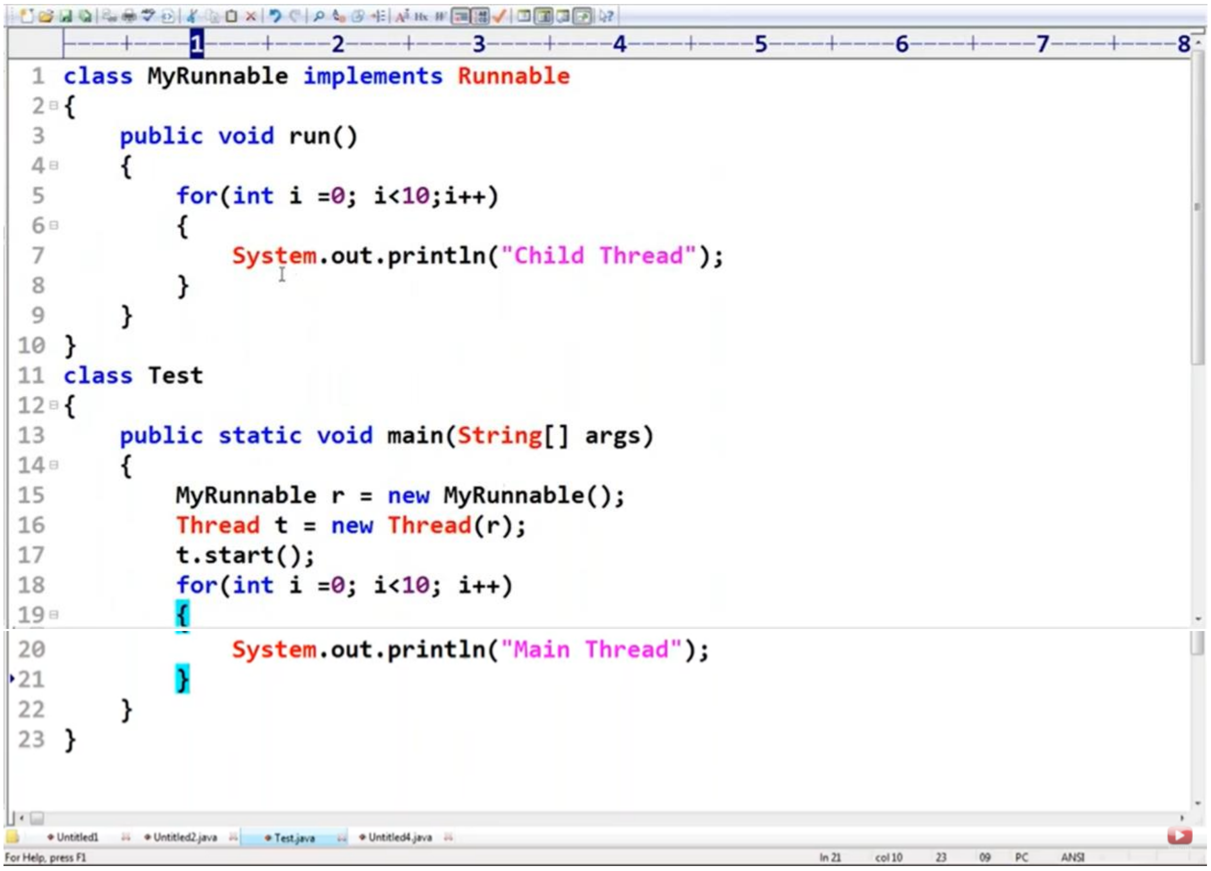
(a,b) are args to add method as my reference is Interf. This is a lambda expression, implementing functional interface that contains 1 abstract method



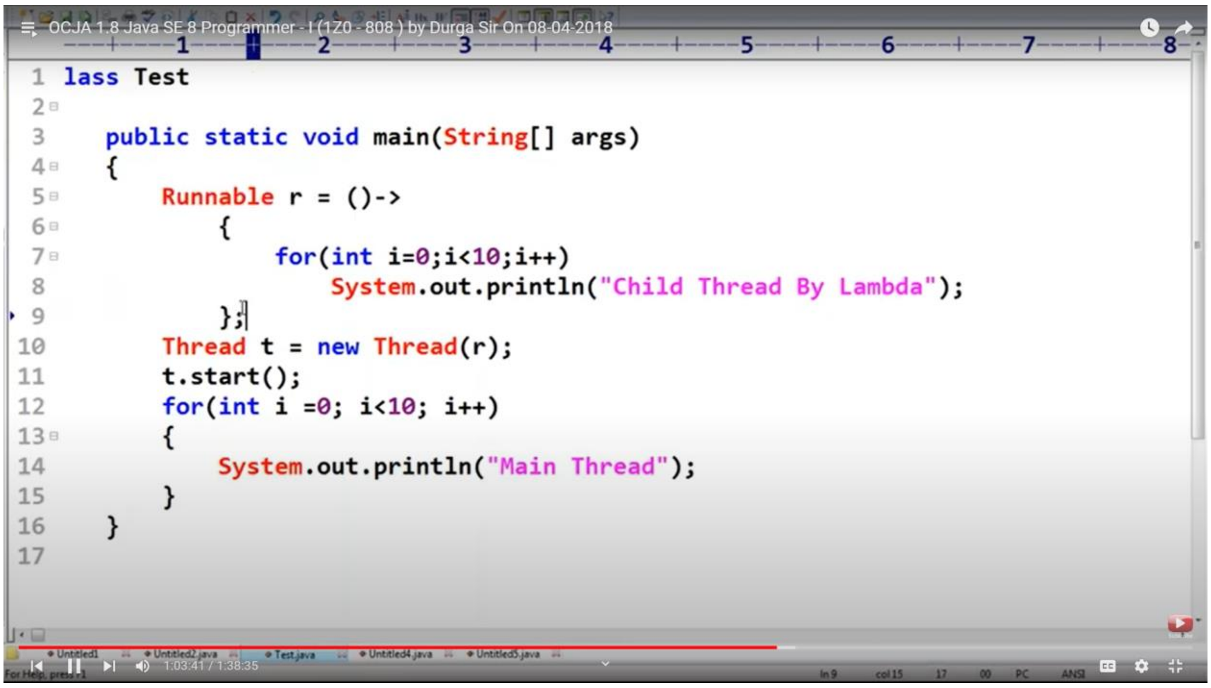
```
interface Interf
{
    public void add(int a,int b);
}
class Test
{
    public static void main(String[] args)
    {
        Interf i=(a,b)->System.out.println("The Sum:"+(a+b));
        i.add(10,20);
        i.add(100,200);
        i.add(1000,2000);
        i.add(10000,20000);
    }
}
```



```
interface Interf
{
    public int squareIt(int x);
}
class Test
{
    public static void main(String[] args)
    {
        Interf i=x->x*x;
        System.out.println(i.squareIt(10));
        System.out.println(i.squareIt(20));
        System.out.println(i.squareIt(30));

    }
}
```

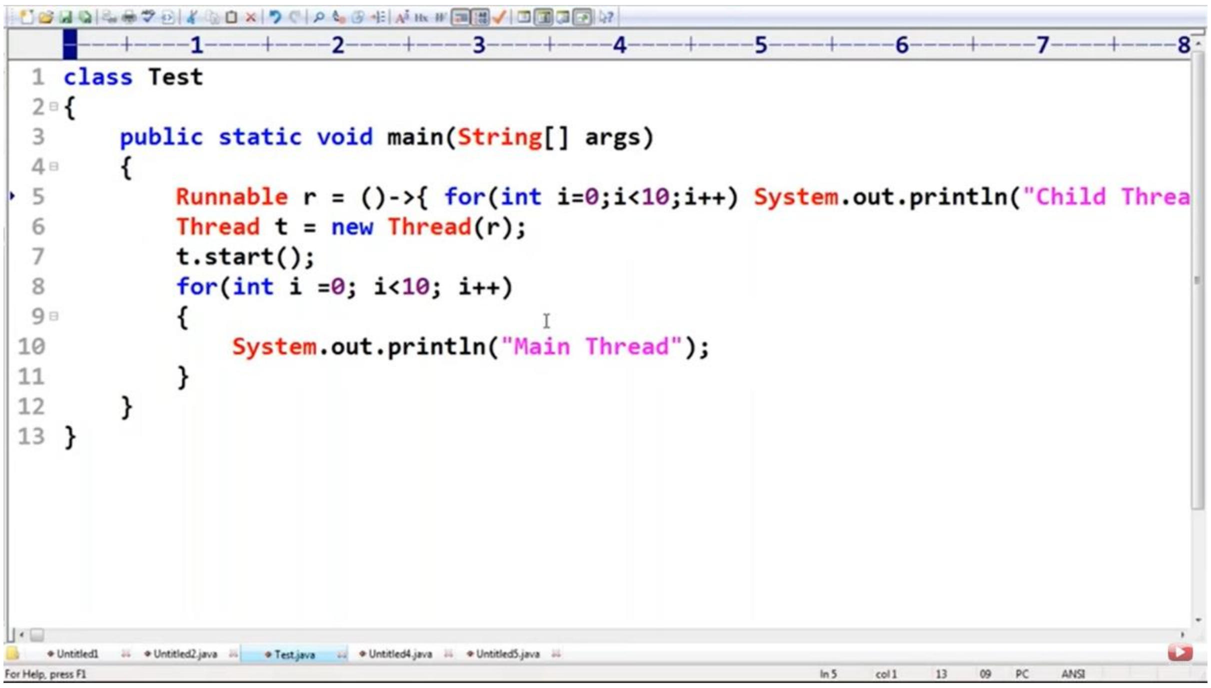


```java
class MyRunnable implements Runnable
{
    public void run()
    {
        for(int i =0; i<10;i++)
        {
            System.out.println("Child Thread");
        }
    }
}
class Test
{
    public static void main(String[] args)
    {
        MyRunnable r = new MyRunnable();
        Thread t = new Thread(r);
        t.start();
        for(int i =0; i<10; i++)
        {
            System.out.println("Main Thread");
        }
    }
}
```



OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018

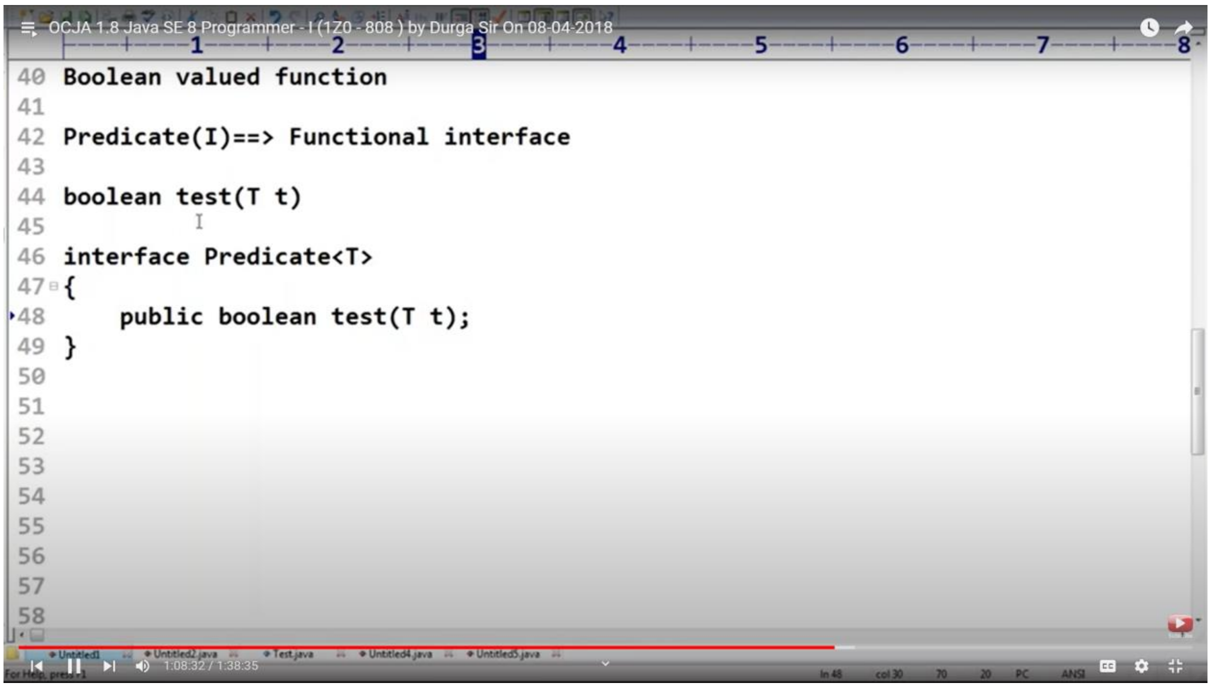
```java
lass Test

  public static void main(String[] args)
  {
      Runnable r = ()->
          {
              for(int i=0;i<10;i++)
                  System.out.println("Child Thread By Lambda");
          };
      Thread t = new Thread(r);
      t.start();
      for(int i =0; i<10; i++)
      {
          System.out.println("Main Thread");
      }
  }
```



```
class Test
{
    public static void main(String[] args)
    {
        Runnable r = ()->{ for(int i=0;i<10;i++) System.out.println("Child Threa
        Thread t = new Thread(r);
        t.start();
        for(int i =0; i<10; i++)
        {
            System.out.println("Main Thread");
        }
    }
}
```

Untitled1 Untitled2.java Test.java Untitled4.java Untitled5.java

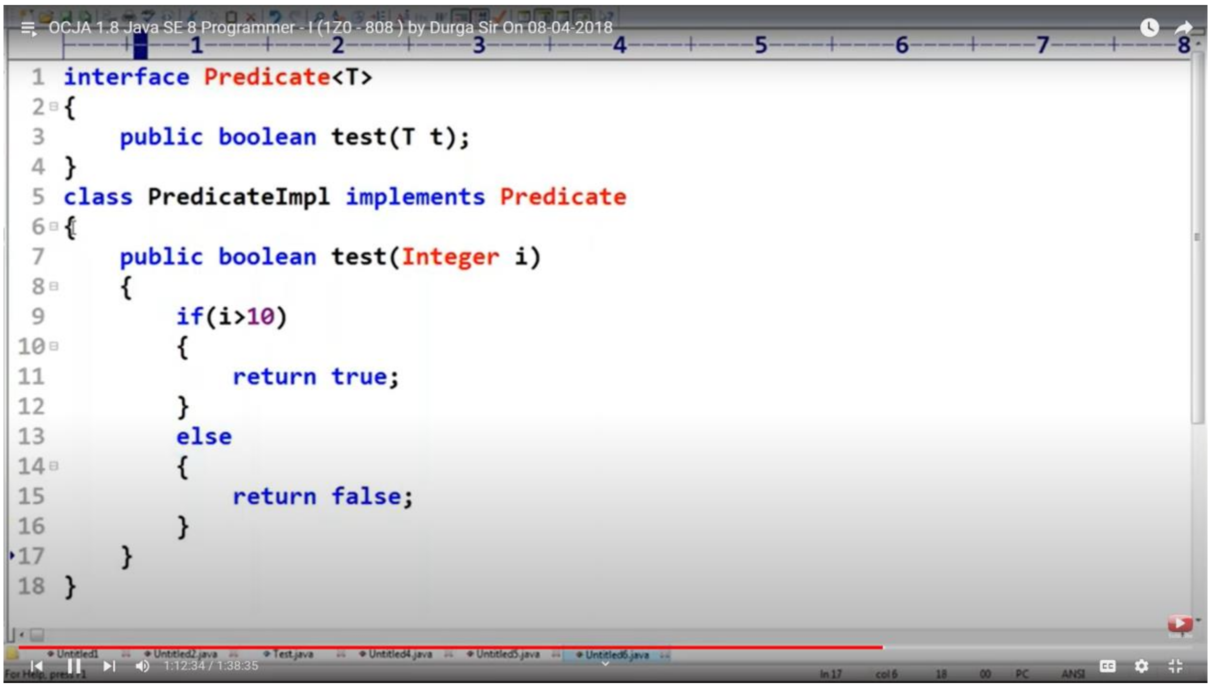
For Help, press F1



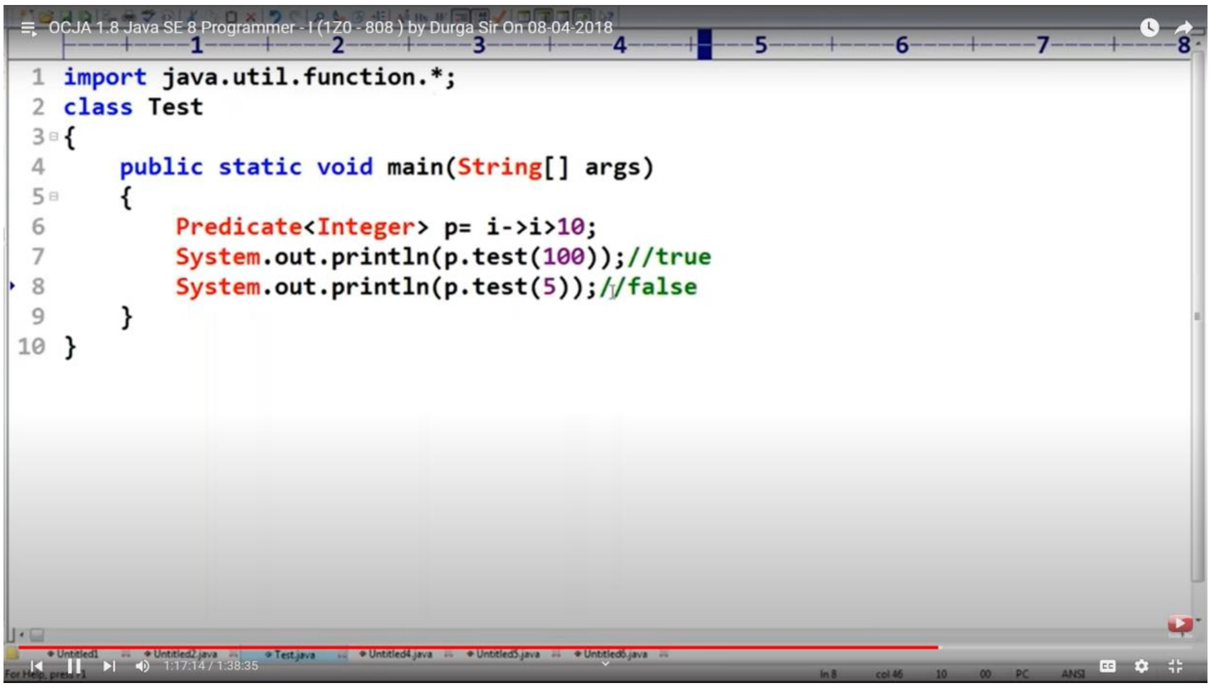
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018

```
Boolean valued function

Predicate(I)==> Functional interface

boolean test(T t)

interface Predicate<T>
{
    public boolean test(T t);
}
```

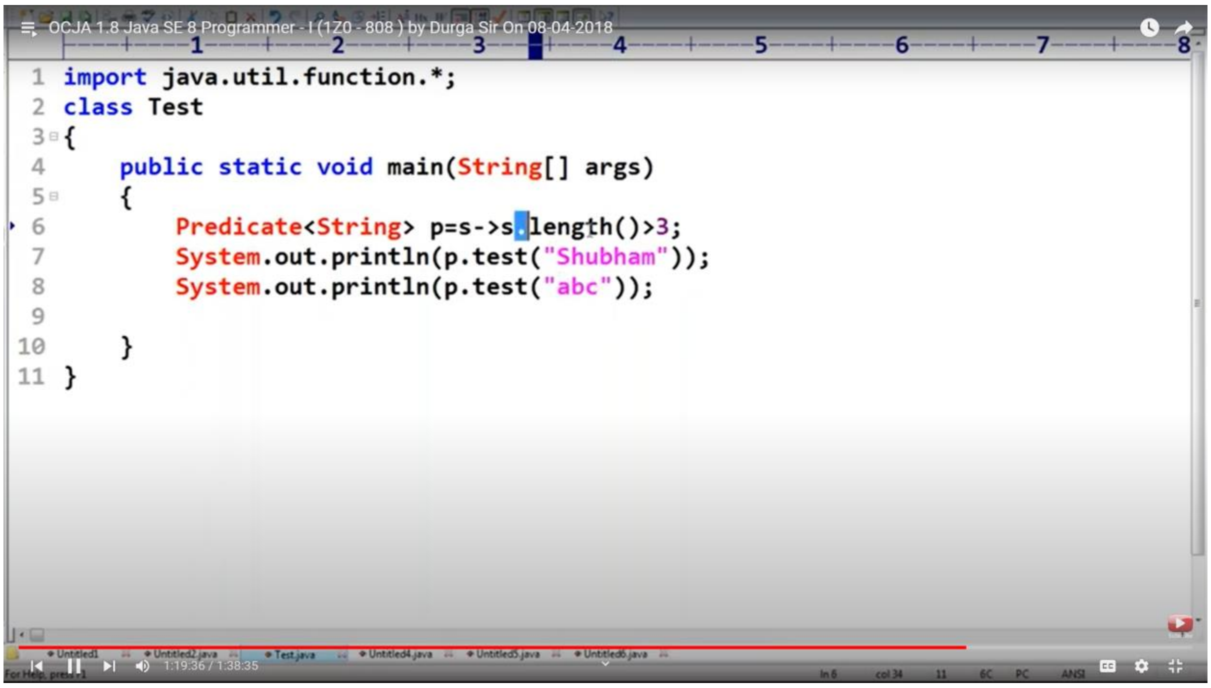
1:08:32 / 1:38:35


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
1:12:34 / 1:38:35

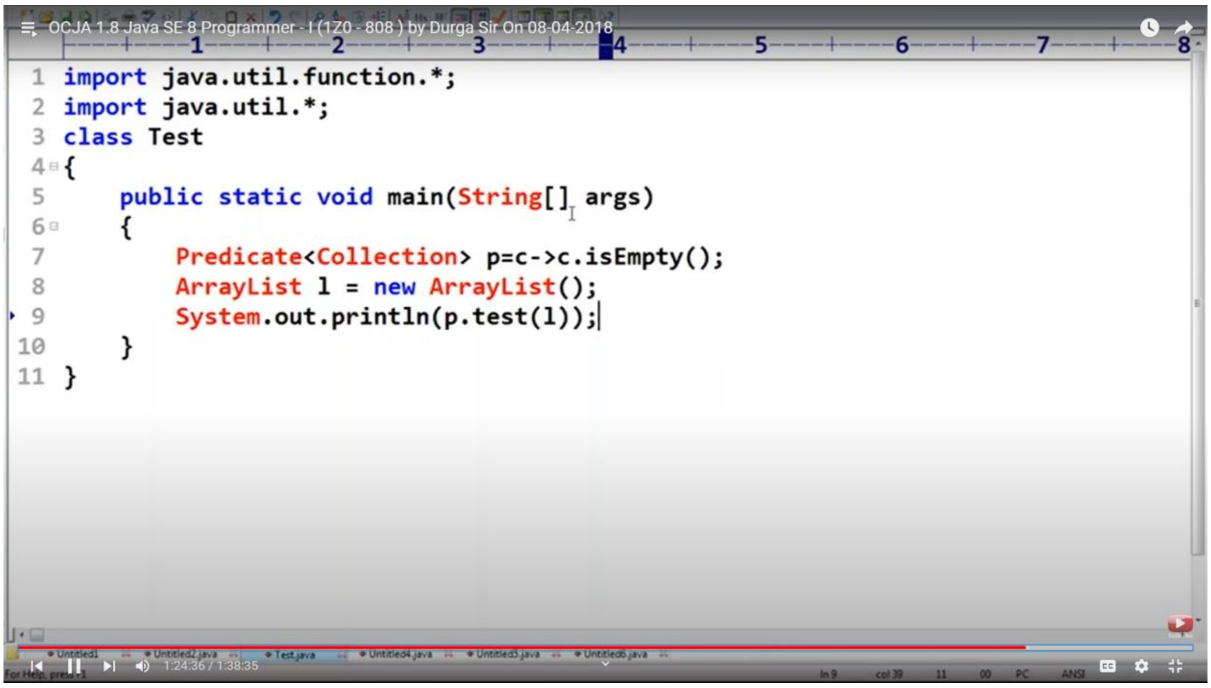
```java
interface Predicate<T>
{
    public boolean test(T t);
}
class PredicateImpl implements Predicate
{
    public boolean test(Integer i)
    {
        if(i>10)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
1:17:14 / 1:38:35

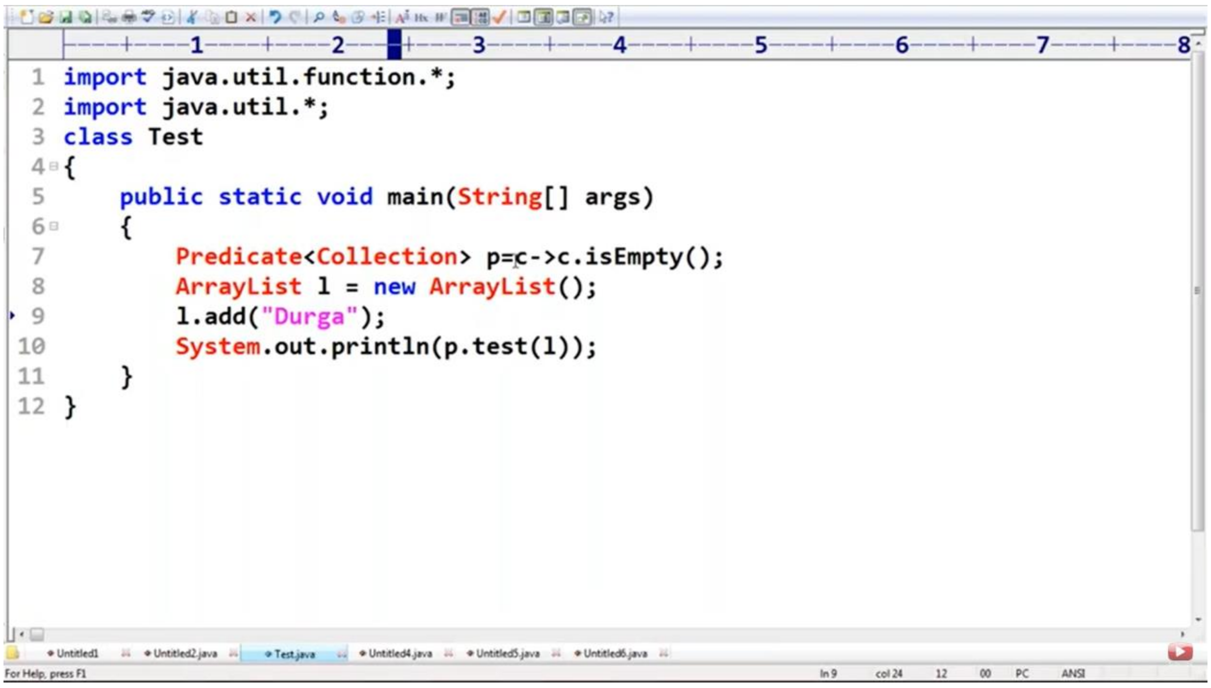
```java
import java.util.function.*;
class Test
{
    public static void main(String[] args)
    {
        Predicate<Integer> p= i->i>10;
        System.out.println(p.test(100));//true
        System.out.println(p.test(5));//false
    }
}
```


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
1:19:36 / 1:38:35

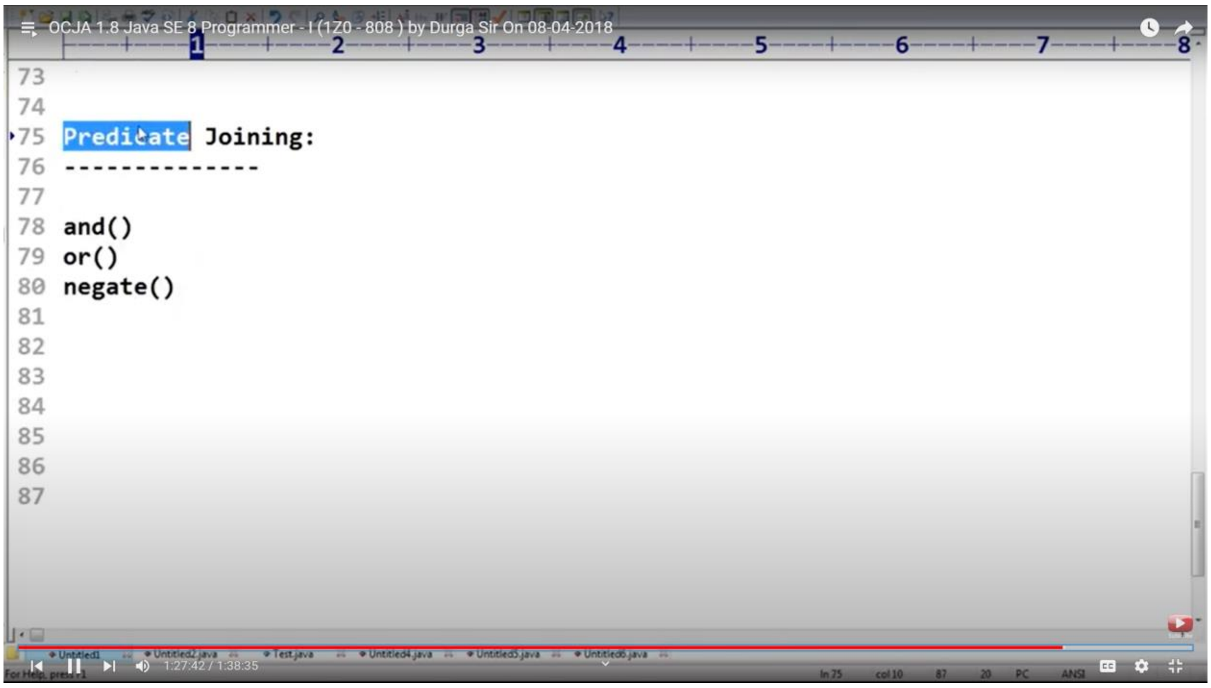
```java
import java.util.function.*;
class Test
{
    public static void main(String[] args)
    {
        Predicate<String> p=s->s.length()>3;
        System.out.println(p.test("Shubham"));
        System.out.println(p.test("abc"));

    }
}
```


OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
1:24:36 / 1:38:35

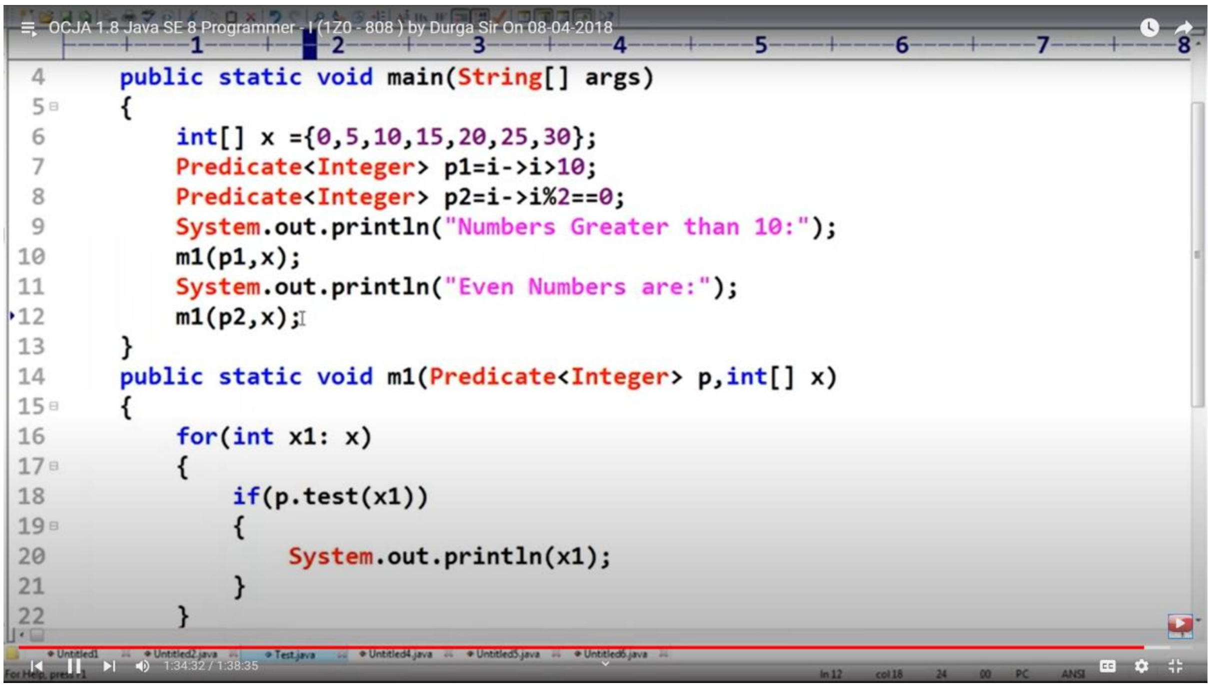
```java
import java.util.function.*;
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Predicate<Collection> p=c->c.isEmpty();
        ArrayList l = new ArrayList();
        System.out.println(p.test(l));
    }
}
```



```
import java.util.function.*;
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Predicate<Collection> p=c->c.isEmpty();
        ArrayList l = new ArrayList();
        l.add("Durga");
        System.out.println(p.test(l));
    }
}
```

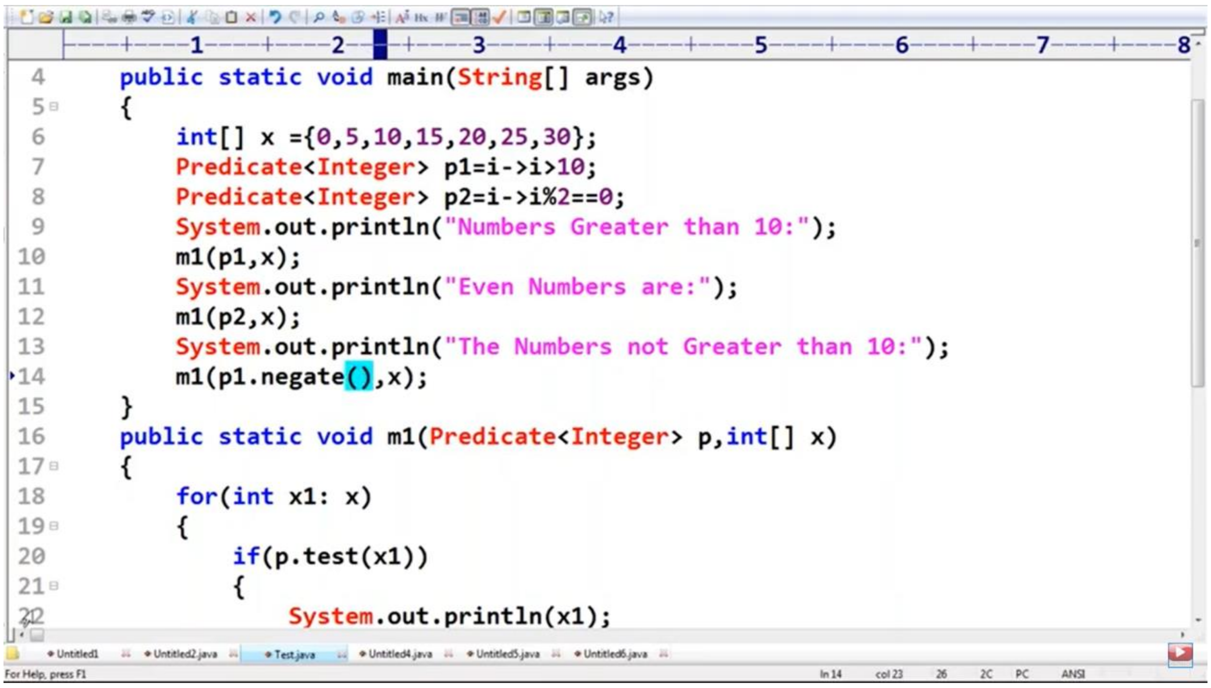

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018
1:27:42 / 1:38:35

```
Predicate Joining:
--------------

and()
or()
negate()
```

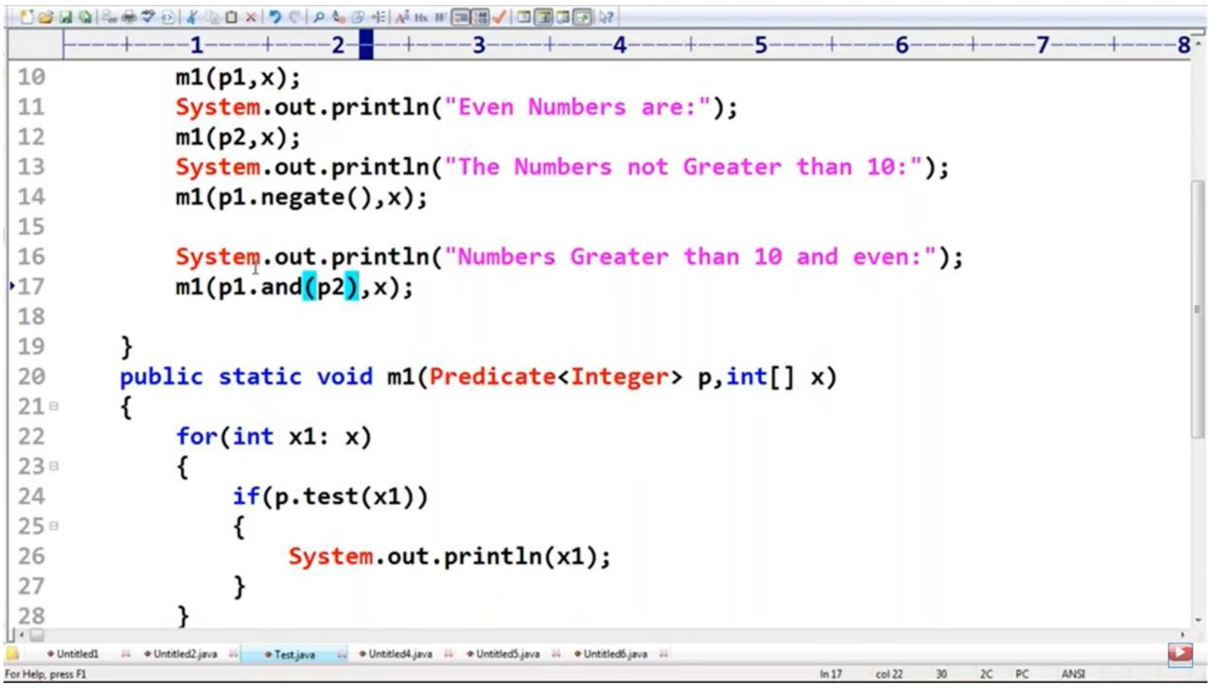

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 08-04-2018

```
    public static void main(String[] args)
    {
        int[] x ={0,5,10,15,20,25,30};
        Predicate<Integer> p1=i->i>10;
        Predicate<Integer> p2=i->i%2==0;
        System.out.println("Numbers Greater than 10:");
        m1(p1,x);
        System.out.println("Even Numbers are:");
        m1(p2,x);
    }
    public static void m1(Predicate<Integer> p,int[] x)
    {
        for(int x1: x)
        {
            if(p.test(x1))
            {
                System.out.println(x1);
            }
        }
```

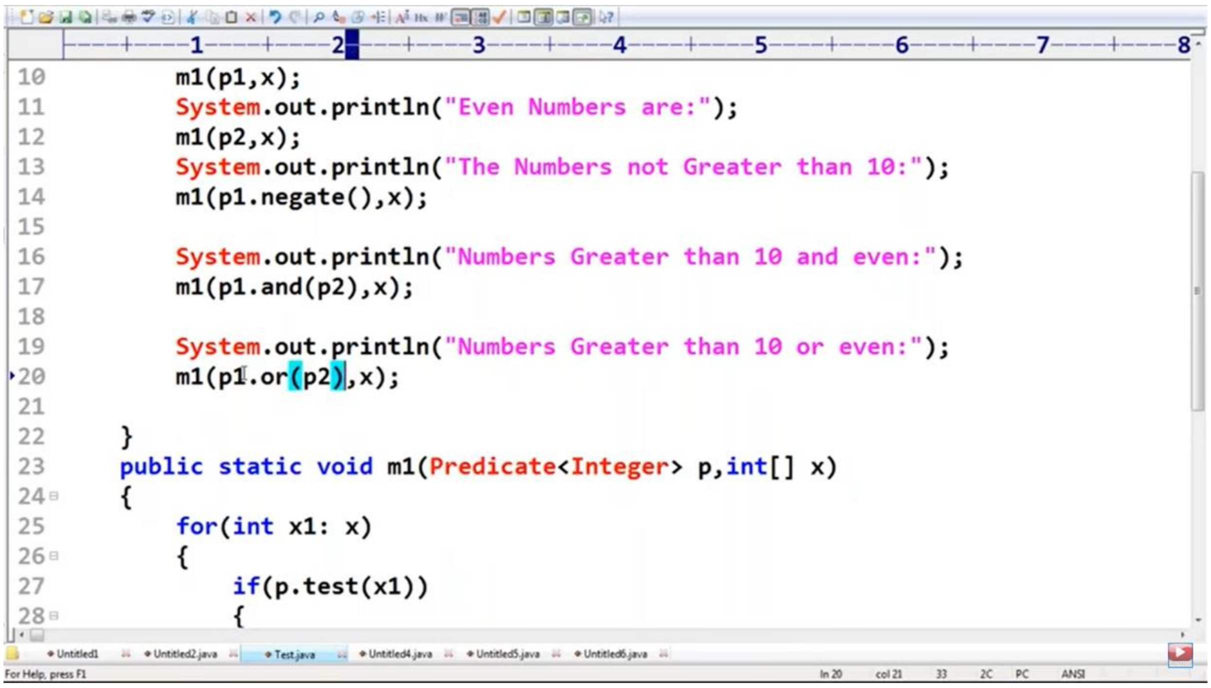
To print nos not greater than 10



```
    public static void main(String[] args)
    {
        int[] x ={0,5,10,15,20,25,30};
        Predicate<Integer> p1=i->i>10;
        Predicate<Integer> p2=i->i%2==0;
        System.out.println("Numbers Greater than 10:");
        m1(p1,x);
        System.out.println("Even Numbers are:");
        m1(p2,x);
        System.out.println("The Numbers not Greater than 10:");
        m1(p1.negate(),x);
    }
    public static void m1(Predicate<Integer> p,int[] x)
    {
        for(int x1: x)
        {
            if(p.test(x1))
            {
                System.out.println(x1);
```



```
        m1(p1,x);
        System.out.println("Even Numbers are:");
        m1(p2,x);
        System.out.println("The Numbers not Greater than 10:");
        m1(p1.negate(),x);

        System.out.println("Numbers Greater than 10 and even:");
        m1(p1.and(p2),x);

    }
    public static void m1(Predicate<Integer> p,int[] x)
    {
        for(int x1: x)
        {
            if(p.test(x1))
            {
                System.out.println(x1);
            }
        }
```



```
        m1(p1,x);
        System.out.println("Even Numbers are:");
        m1(p2,x);
        System.out.println("The Numbers not Greater than 10:");
        m1(p1.negate(),x);

        System.out.println("Numbers Greater than 10 and even:");
        m1(p1.and(p2),x);

        System.out.println("Numbers Greater than 10 or even:");
        m1(p1.or(p2),x);

    }
    public static void m1(Predicate<Integer> p,int[] x)
    {
        for(int x1: x)
        {
            if(p.test(x1))
            {
```