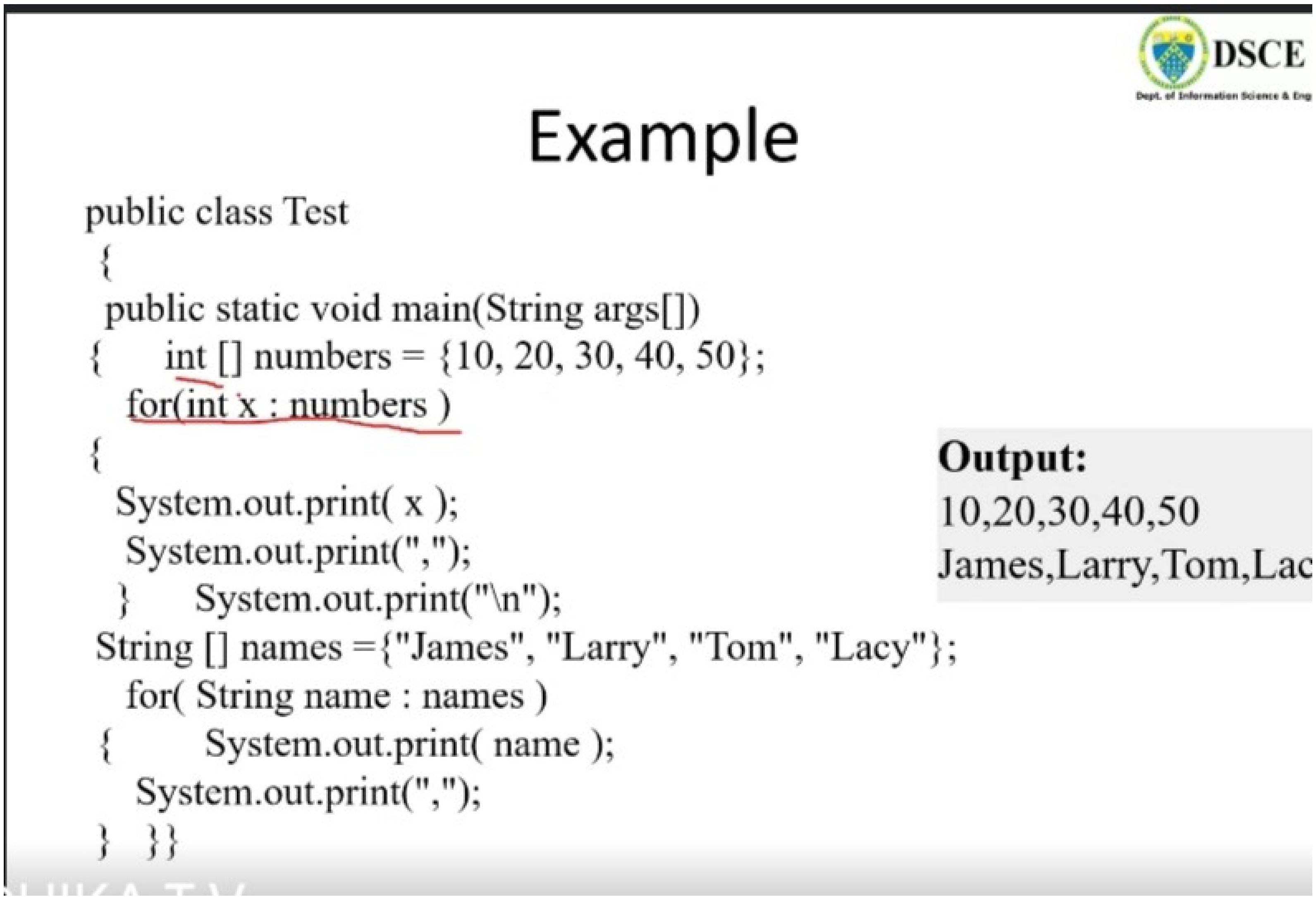

DSCE

Dept. of Information Science & Eng

Example

```
public class Test
 {
 public static void main(String args[])
{     int [] numbers = {10, 20, 30, 40, 50};
   for(int x : numbers )
{
  System.out.print( x );
   System.out.print(",");
  }     System.out.print("\n");
 String [] names ={"James", "Larry", "Tom", "Lacy"};
   for( String name : names )
 {        System.out.print( name );
    System.out.print(",");
 }  }}
```

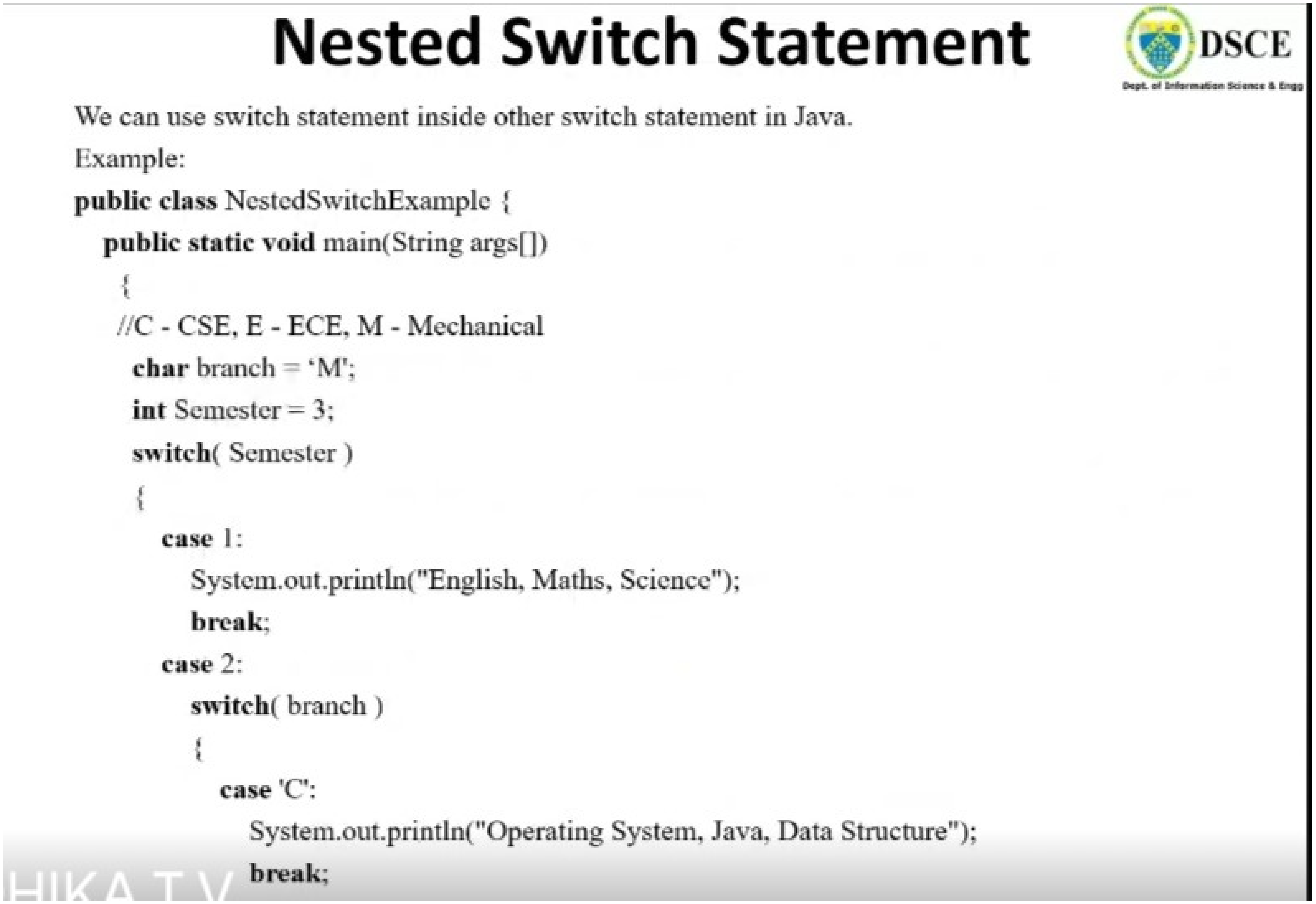
Output:

```
10,20,30,40,50
James,Larry,Tom,Lac
```

The break Keyword:

- The *break* keyword is used to stop the entire loop.
- The break keyword must be used inside any loop or a switch statement.
- The break keyword will stop the execution of the innermost loop and start executing the next line of code after the block.
- The syntax of a break is a single statement inside any loop:

```
break;
```



Nested Switch Statement

DSCE
Dept. of Information Science & Engg

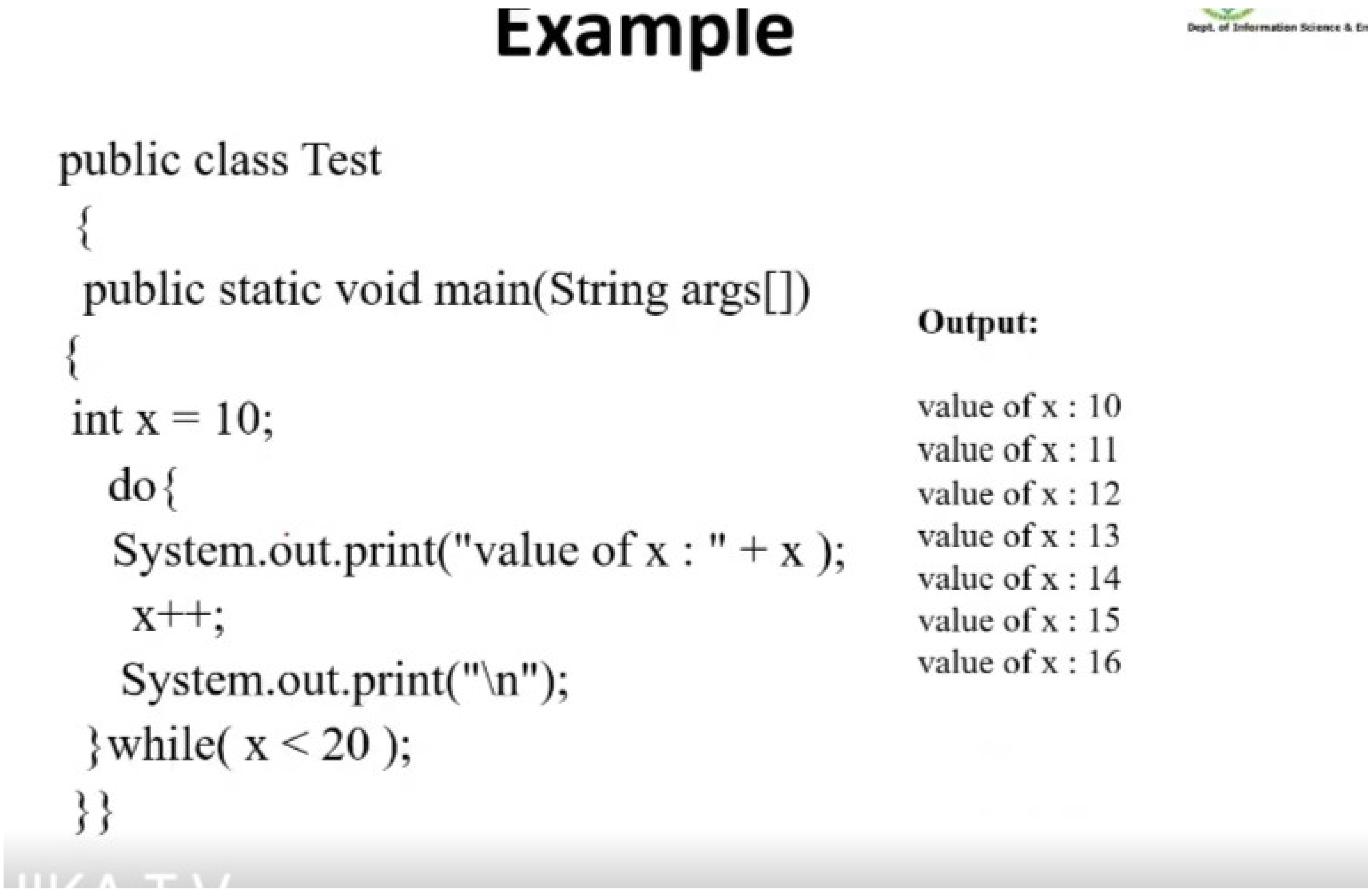
We can use switch statement inside other switch statement in Java.

Example:

```
public class NestedSwitchExample {
   public static void main(String args[])
    {
    //C - CSE, E - ECE, M - Mechanical
      char branch = ‘M’;
      int Semester = 3;
      switch( Semester )
       {
         case 1:
            System.out.println("English, Maths, Science");
            break;
         case 2:
            switch( branch )
            {
               case 'C':
                  System.out.println("Operating System, Java, Data Structure");
                  break;
```

Cntd..

```
  case 'E':
           System.out.println("Micro processors, Logic switching theory");
           break;
         case 'M':
           System.out.println("Drawing, Manufacturing Machines");
           break;
       }  break;
case 3:
       switch( branch )
       {
         case 'C':
           System.out.println("Computer Organization, MultiMedia");
           break;
         case 'E':
           System.out.println("Fundamentals of Logic Design, Microelectronics");
           break;
         case 'M':
           System.out.println("Internal Combustion Engines, Mechanical Vibration");
           break;
       }
       break;
```

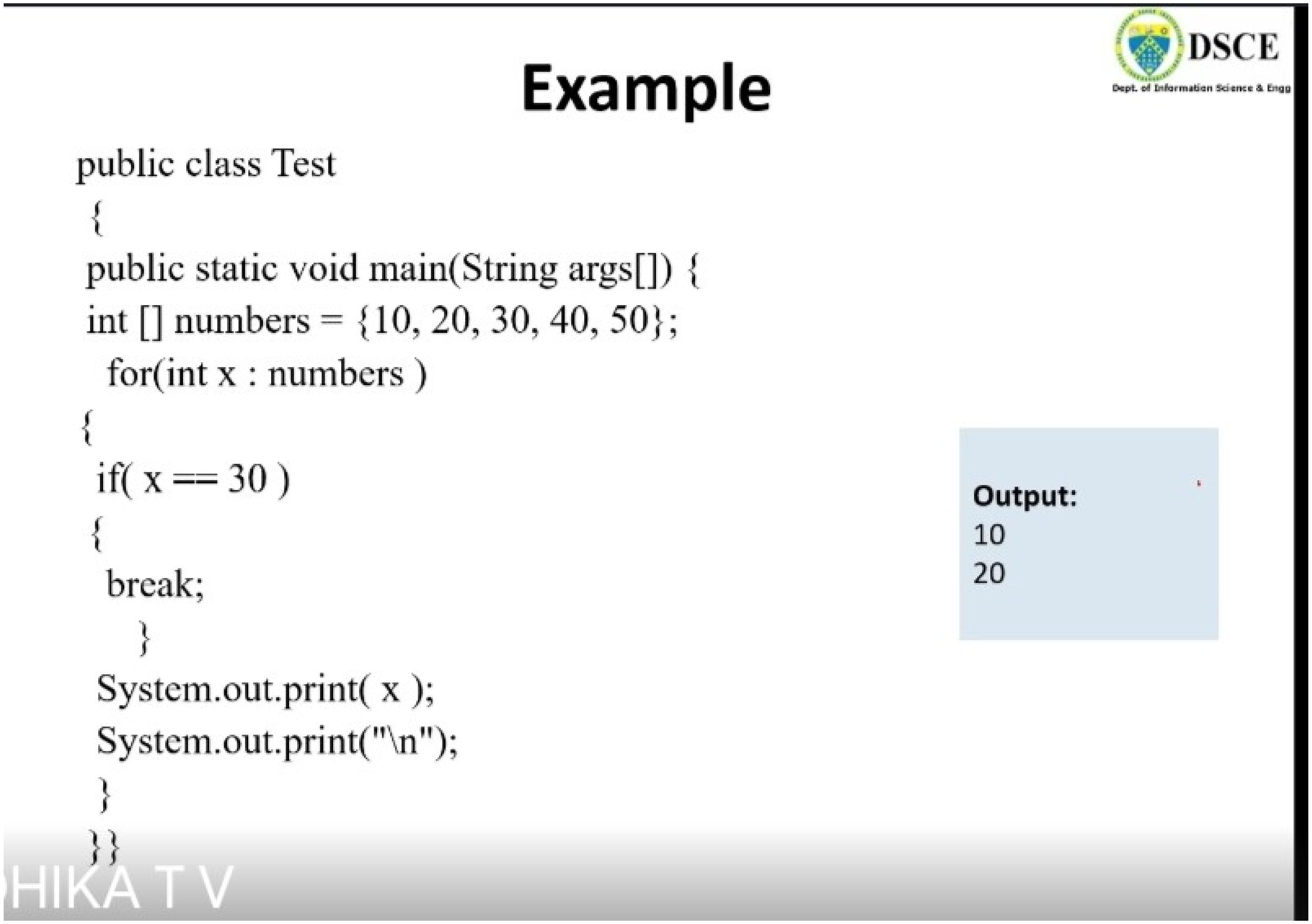


Example

```
public class Test
 {
  public static void main(String args[])
{
 int x = 10;
    do{
    System.out.print("value of x : " + x );
     x++;
     System.out.print("\n");
  }while( x < 20 );
 }}
```

Output:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
```



DSCE
Dept. of Information Science & Engg

Example

```
public class Test
 {
 public static void main(String args[]) {
 int [] numbers = {10, 20, 30, 40, 50};
   for(int x : numbers )
{
  if( x == 30 )
 {
   break;
      }
  System.out.print( x );
  System.out.print("\n");
 }
 }}
```

Output:

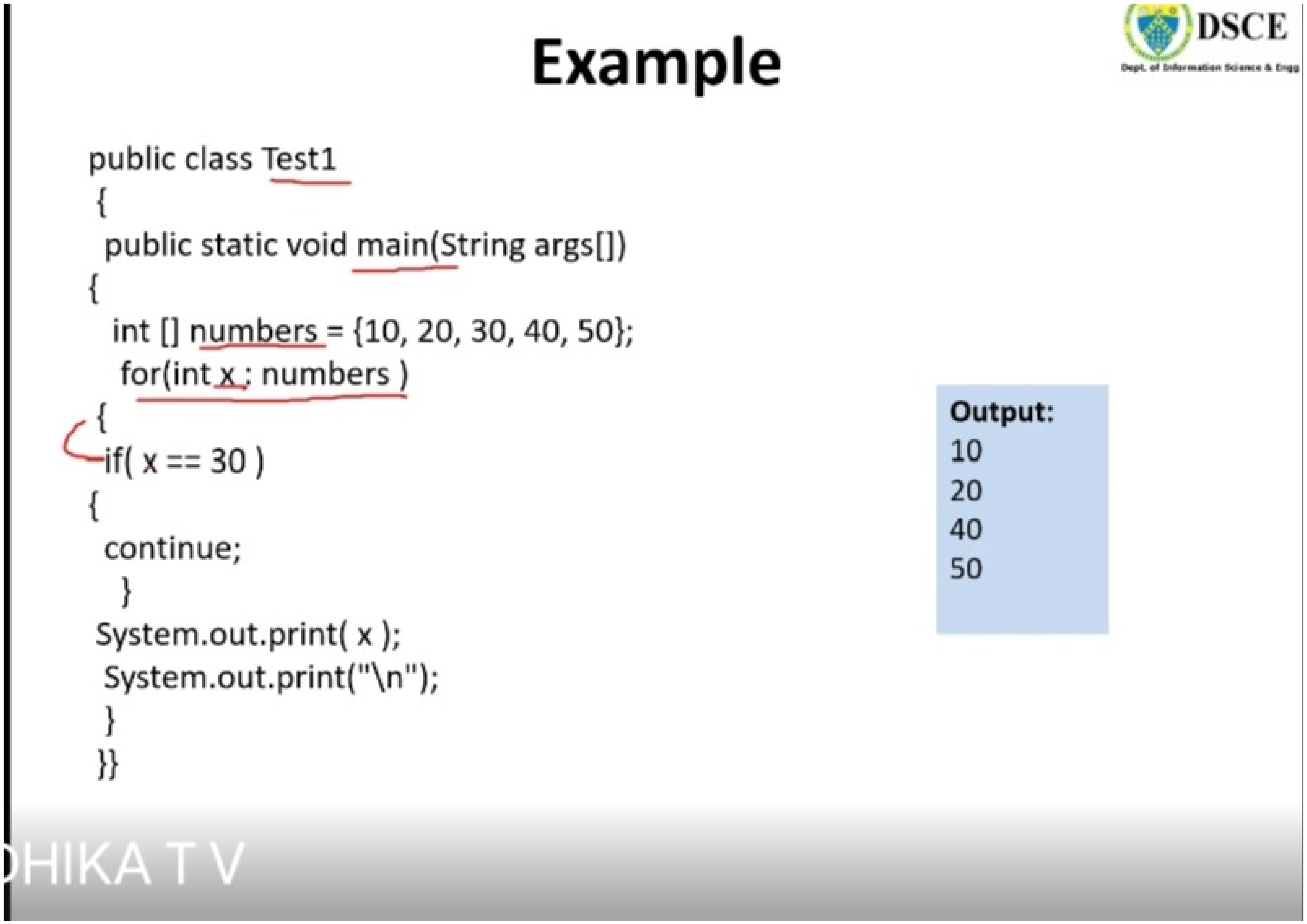
```
10
20
```

HIKA T V

The continue Keyword:

- The *continue* keyword can be used in any of the loop control structures.
- It causes the loop to immediately jump to the next iteration of the loop.
- In a for loop, the continue keyword causes flow of control to immediately jump to the update statement.
- In a while loop or do/while loop, flow of control immediately jumps to the Boolean expression.
- The syntax of a continue is a single statement inside any loop:

  continue;



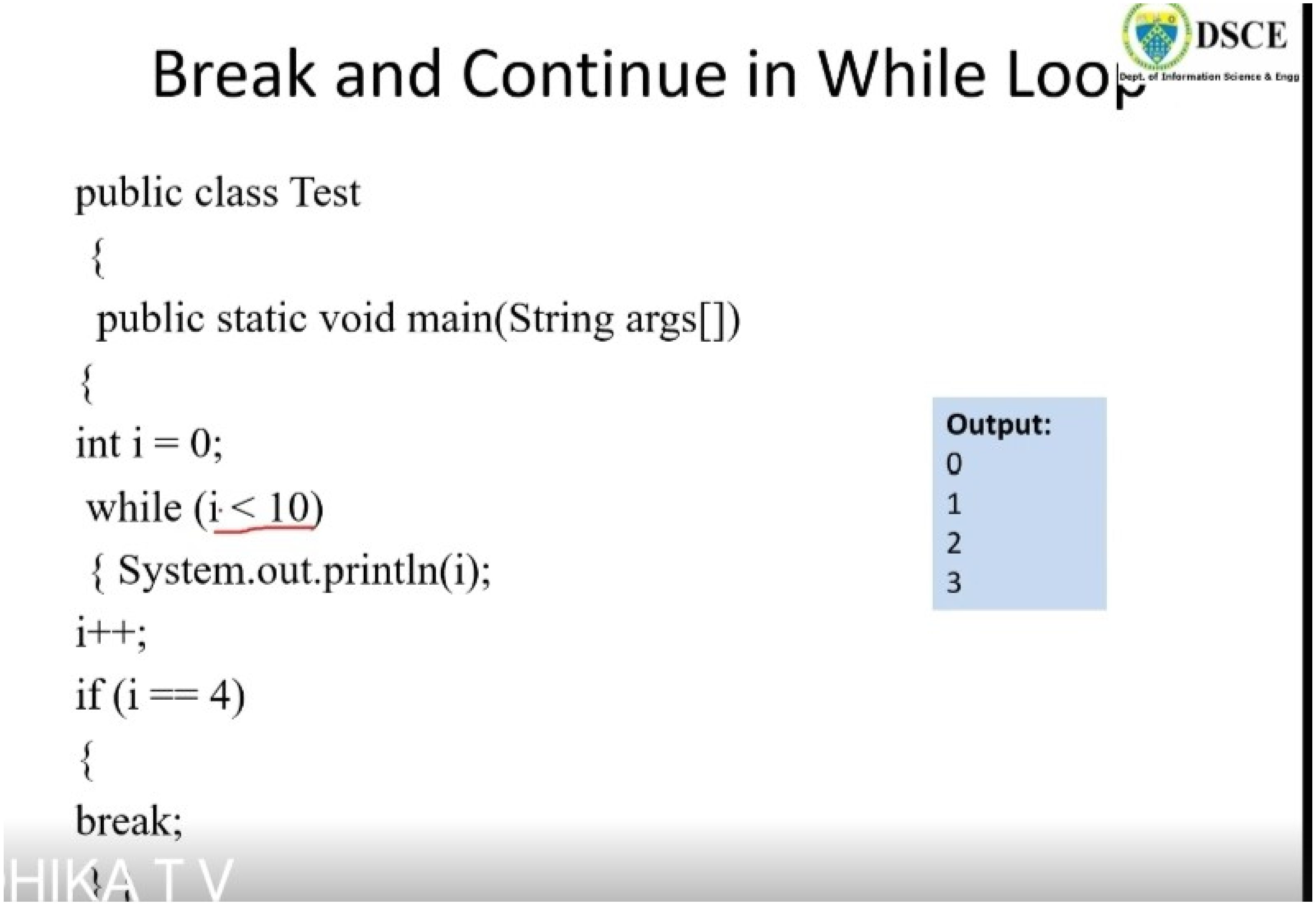
DSCE
Dept. of Information Science & Engg

Example

```
public class Test1
 {
  public static void main(String args[])
 {
   int [] numbers = {10, 20, 30, 40, 50};
    for(int x : numbers )
  {
 if( x == 30 )
 {
  continue;
    }
 System.out.print( x );
  System.out.print("\n");
  }
 }}
```

Output:

```
10
20
40
50
```



Break and Continue in While Loop

```
public class Test
 {
  public static void main(String args[])
 {
int i = 0;
 while (i < 10)
 { System.out.println(i);
i++;
if (i == 4)
{
break;
 }
```

Output:

```
0
1
2
3
```

Break and Continue in While Loo

DSCE
Dept. of Information Science & Engg

```
public class Test
 {
  public static void main(String args[])
{
int i = 0;
 while (i < 10)
{
if (i == 4)
{
i++;
continue;
}
 System.out.println(i);
i++;
} } }
```

```
Output:
0
1
2
3
5
6
7
8
9
```

HIKA T V

DSCE
Dept. of Information Science & Engg

Class in JAVA

- A class is a user defined blueprint or prototype from which objects are created. Objects are real life entities(or) it is an instance of class.
- In general, class declarations can include following components, in order:
- **Modifiers** : A class can be public or has default access
- **Class name:**The name should begin with Capitial letter
- **Superclass(if any):**The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- **Interfaces(if any):**A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- **Body:**The class body surrounded by braces, { }.

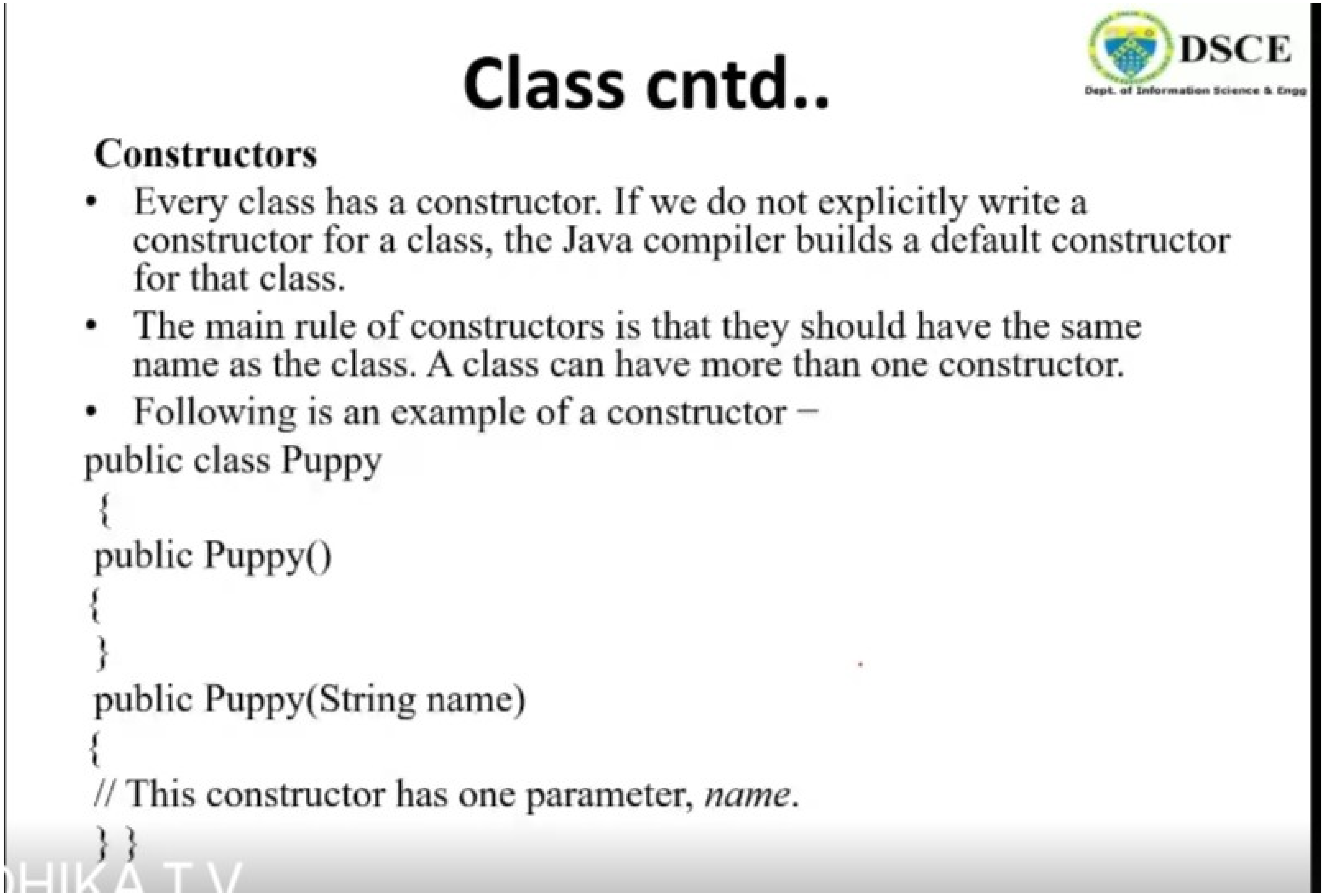
DSCE
Dept. of Information Science & Engg

Class cntd..

- An object consists of :
- **State**: It is represented by attributes of an object. It also reflects the properties of an object.
- **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.
- **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Example of Class

```
public class Dog
{
 String breed;
int age;
String color;
void barking()
{
 }
 void hungry()
 {
}
void sleeping()
{
 }
 }
```

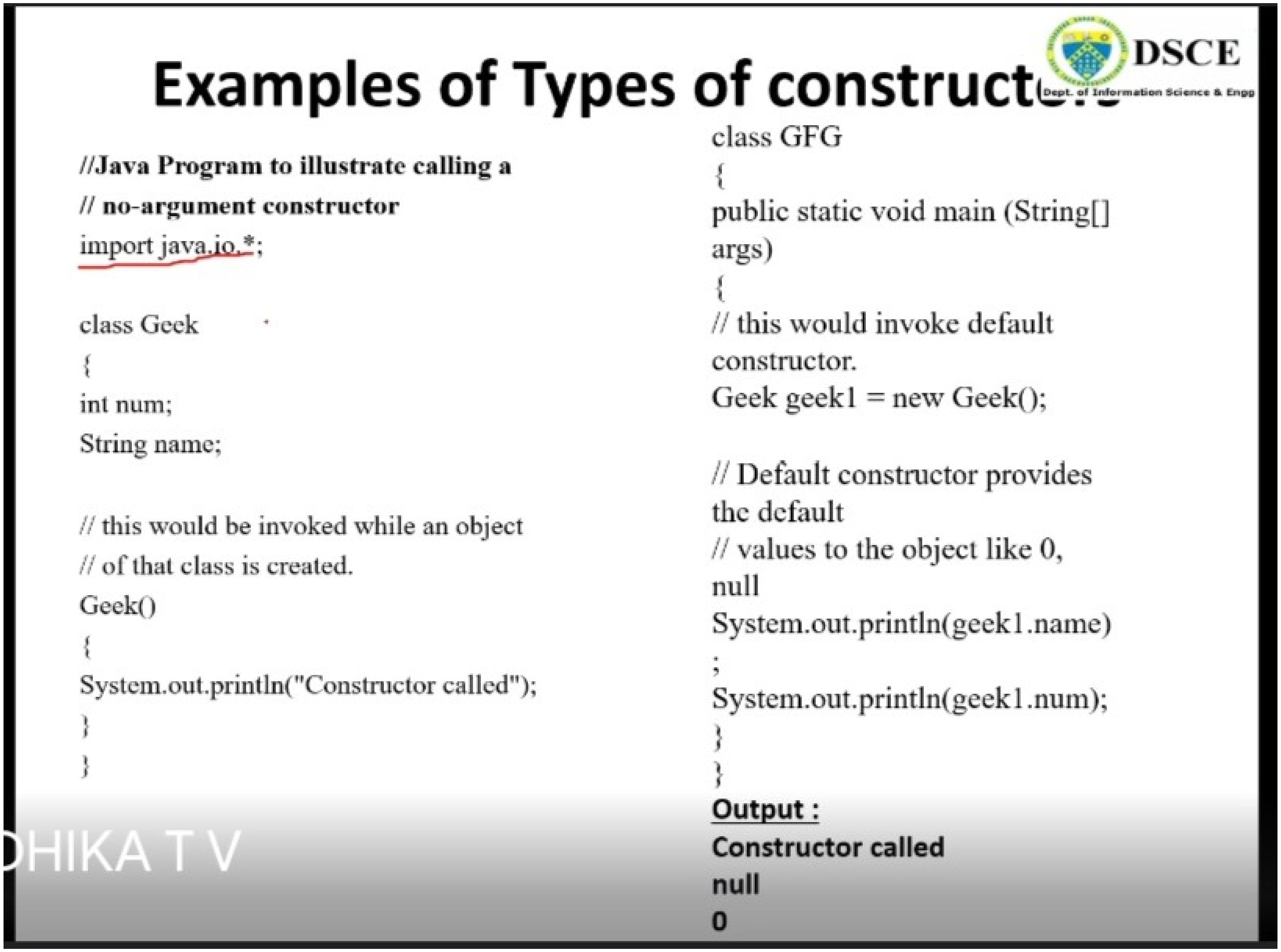


Class cntd..

Constructors

- Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.
- The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.
- Following is an example of a constructor −

```
public class Puppy
 {
 public Puppy()
{
 }
 public Puppy(String name)
{
 // This constructor has one parameter, name.
 } }
```



Examples of Types of construct

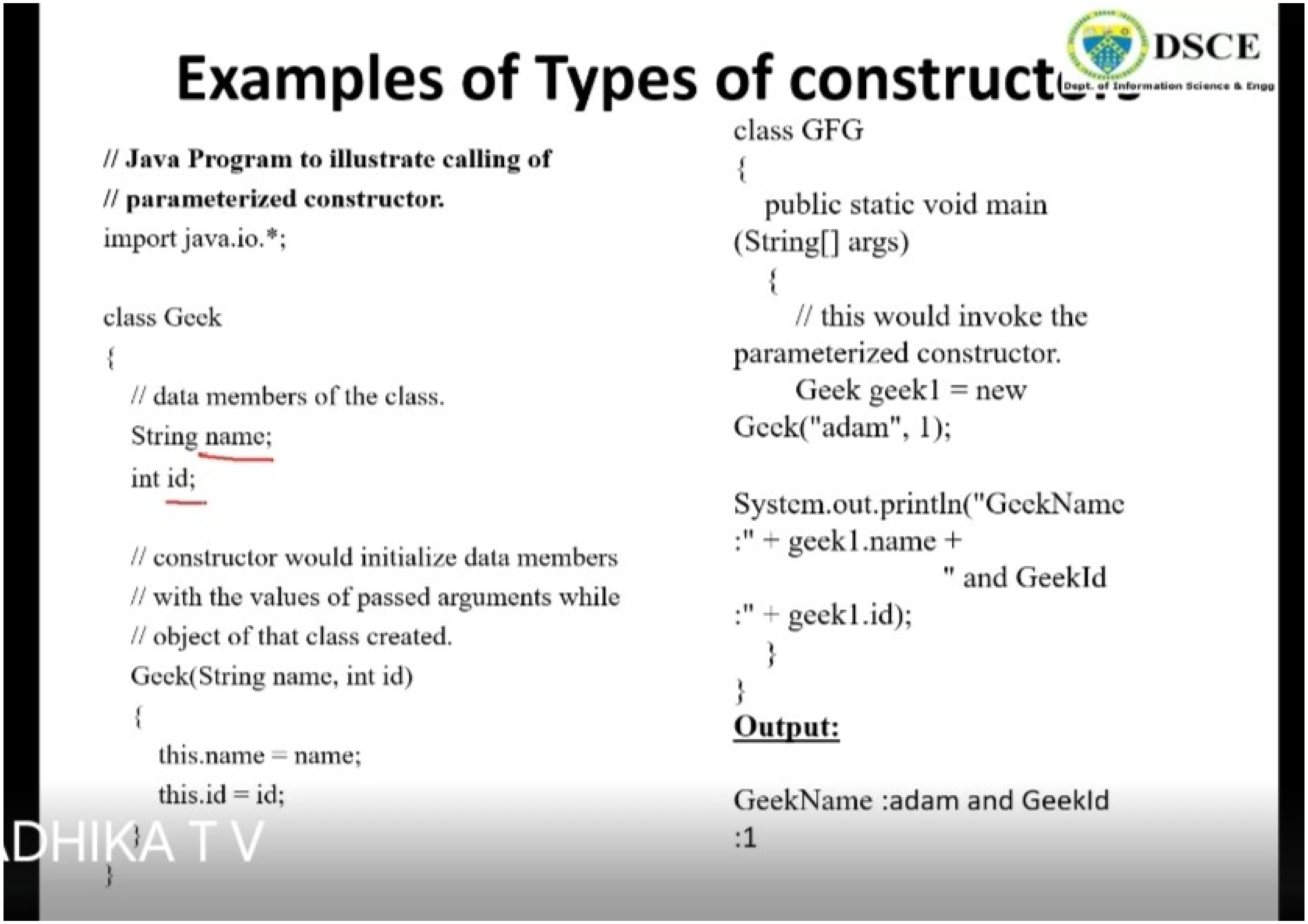
DSCE
Dept. of Information Science & Engg

```
//Java Program to illustrate calling a
// no-argument constructor
import java.io.*;

class Geek
{
int num;
String name;

// this would be invoked while an object
// of that class is created.
Geek()
{
System.out.println("Constructor called");
}
}
```

```
class GFG
{
public static void main (String[]
args)
{
// this would invoke default
constructor.
Geek geek1 = new Geek();

// Default constructor provides
the default
// values to the object like 0,
null
System.out.println(geek1.name)
;
System.out.println(geek1.num);
}
}
```

Output :

```
Constructor called
null
0
```



DSCE
Dept. of Information Science & Engg

Examples of Types of constructors

```
// Java Program to illustrate calling of
// parameterized constructor.
import java.io.*;

class Geek
{
   // data members of the class.
   String name;
   int id;

   // constructor would initialize data members
   // with the values of passed arguments while
   // object of that class created.
   Geek(String name, int id)
   {
      this.name = name;
      this.id = id;
   }
}
```

```
class GFG
{
   public static void main
(String[] args)
   {
      // this would invoke the
parameterized constructor.
      Geek geek1 = new
Geek("adam", 1);

System.out.println("GeekName
:" + geek1.name +
                       " and GeekId
:" + geek1.id);
   }
}
```

Output:

```
GeekName :adam and GeekId
:1
```

Class cntd..

Creating an Object

- A class provides the blueprints for objects. So basically, an object is created from a class.
- In Java, the new keyword is used to create new objects.
- There are three steps when creating an object from a class −
- **Declaration** − A variable declaration with a variable name with an object type.
- **Instantiation** − The 'new' keyword is used to create the object.
- **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.