Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

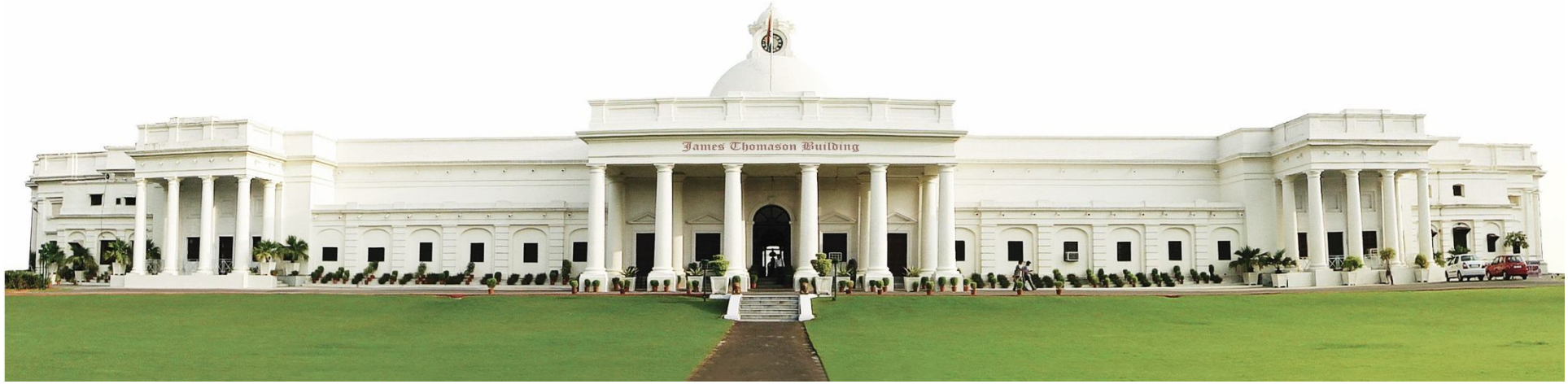
Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/



Method returning Objects

```java
class distance{
        int feet;
        int inches;
        distance(int x , int y)
        {
                feet=x;
                inches=y;
        }
        void displaydistance()
        {
                System.out.println(feet+" feet"+inches+" inchess");
        }
        static distance add (distance one,distance two)
        {
                int f=one.feet+two.feet;
                int i=one.inches+two.inches;
                if(i>=12)
                {
                        f++;
                        i=i-12;
                }
                distance d=new distance(f,i);
                return d;
        }
}//distance type created
```

```java
class Executedistance {

        public static void main(String[] args) {
                distance d1=new distance(10,9);
                System.out.println("the first distance is :");
                d1.displaydistance();
                distance d2=new distance(9,10);
                System.out.println("the second distance is :");
                d2.displaydistance();
                distance sum=distance.add(d1,d2);
                System.out.println("the sum of their distance is :");
                sum.displaydistance();

        }

}
```

Terminal

```
sh-4.3$ javac Executedistance.java
sh-4.3$ java Executedistance
the first distance is :
10 feet 9 inchess
the second distance is :
9 feet 10 inchess
the sum of their distance is :
20 feet 7 inchess
sh-4.3$
```

Using Command-Line Arguments

```java
public class CommandLine {

    public static void main(String args[]){
        for(int i=0; i<args.length; i++){
            System.out.println("args[" + i + "]: " + args[i]);
        }
    }
}
```

Terminal

```
sh-4.3$ javac CommandLine.java
sh-4.3$ java CommandLine All the best for your Mid Term Examinations
args[0]: All
args[1]: the
args[2]: best
args[3]: for
args[4]: your
args[5]: Mid
args[6]: Term
args[7]: Examinations
sh-4.3$
```

Passing Arrays to Methods

```java
import java.util.Scanner;
public class FindSum
{
    public static void main (String [ ] args)
    {
        int [ ]  number = new int [ 10];   // instantiate the array
        int i;
        int sum=0;
        Scanner in = new Scanner(System.in);
        for ( i = 0; i < 10; i++ )            // fill the array
            number[ i ] = in.nextInt();

        sum = find_sum(number);   // invoke the method
        System.out.println("The sum is"+ " " +sum + ".");
    }

    public static int  find_sum(int [ ] value)  //method definition to find sum
    {
        int i, total = 0;
        for(i=0; i<10; i++)
        {
            total = total + value[ i ];
        }

        return total;
    }
}
```

value → @a007

number → 0x23ef | 7 | 3 | 14 | 8 ...

number[0]

value[0] → 7

value[1] → 3

Terminal

```
sh-4.3$ javac FindSum.java
sh-4.3$ java FindSum
10 20 30 40 50 60 70 80 90 100
The sum is 550.
sh-4.3$
```

Java passes reference by value

```java
import java.util.Scanner;
public class Modify
{
    public static void main (String [ ] args)
    {
        int [ ]  number = new int [3];   // instantiate the array
        int i;
        int sum=0;
        Scanner in = new Scanner(System.in);
        System.out.println("Outside method");
        for ( i = 0; i < 3; i++ )            // fill the array
           {number[ i ] = i+1;
           if (i==2)
           System.out.println(" " +number[i]);
           else
           System.out.print(" " +number[i]);
           }
         System.out.println("Reference of an Array number " +number);
        modify_array(number);   // invoke the method

    }

```

& @ a007

1 2 3

#

```java
    public static void  modify_array(int [ ] value)  //method definition to find sum
    {
        int i;
        System.out.println("Inside method");
        for(i=0; i<3; i++)
          {value[ i ]=-10;
          if (i==2)
             System.out.println(" " +value[i]);
          else
          System.out.print(" " +value[i]);
          }
        System.out.println("Reference of an Array value " +value);

    }
}
```

-10 -10 -10

Terminal

```
sh-4.3$ javac Modify.java
sh-4.3$ java Modify
Outside method
 1 2 3
Reference of an Array number [I@5c647e05
Inside method
 -10 -10 -10
Reference of an Array value [I@5c647e05
sh-4.3$
```

WAP using a method to find the number of elements in a given integer array which are divisible by 7. Pass this array in the method.

```java
import java.util.Scanner;
public class FindDiv7
{
     public static void main (String [ ] args)
     {
          int [ ]  number = new int [ 10];   // instantiate the array
          int i;
          Scanner in = new Scanner(System.in);
          for ( i = 0; i < 10; i++ )            // fill the array
             number[ i ] = in.nextInt();

          int count = find_div_by_7(number);   // invoke the method
         System.out.println("The count is"+ " " +count + ".");
     }

    public static int  find_div_by_7(int [ ] value)  //method definition to find sum
    {
         int i, count = 0;
         for(i=0; i<10; i++)
         {  if (value[i]%7==0)
           ++count;
         }

         return count;
    }
}
```

Terminal

```
sh-4.3$ javac FindDiv7.java
sh-4.3$ java FindDiv7
7 14 21 40 50 60 70 80 90 100
The count is 4.
sh-4.3$
```

JAVA

- **Java is pass by value**. Well, **pass by reference value**.

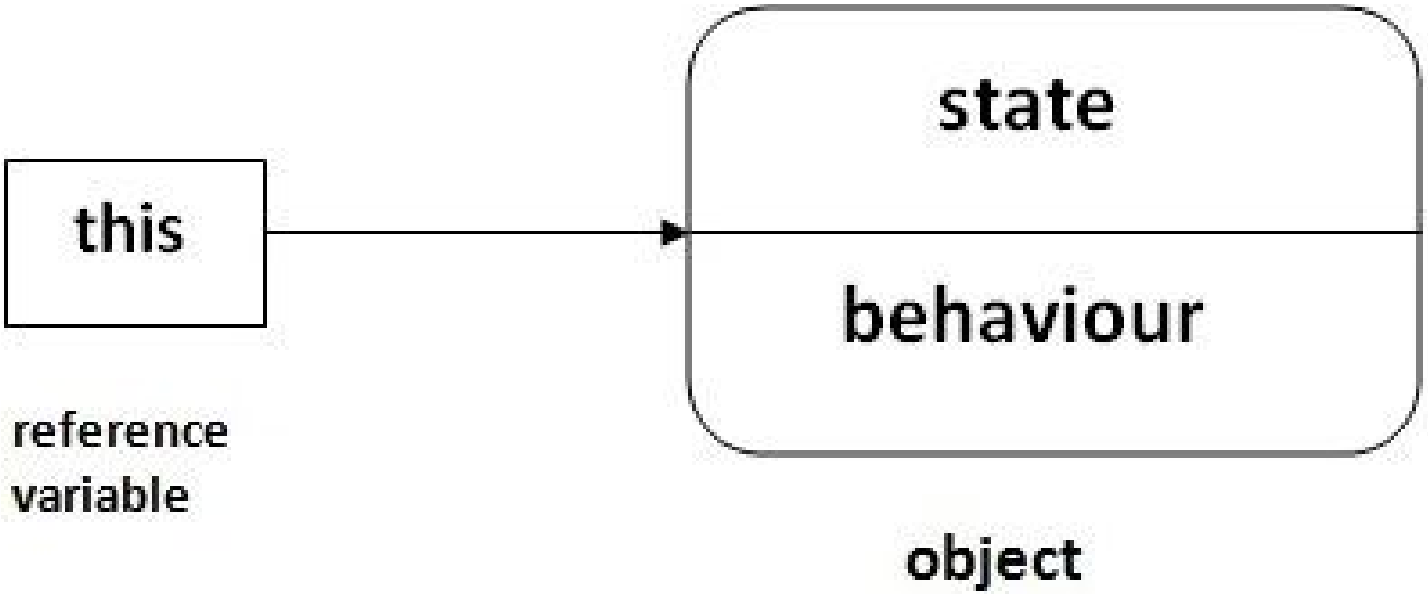
- Oh well, even better is ***pass-by-copy-of-the-variable-value! ;)***

this keyword in java

- In java, this is a **reference variable** that refers to the current object.
- There can be a lot of usage of **java this keyword**.

Usage of java this keyword

1. this keyword can be used to refer current class instance variable.
2. this() can be used to invoke current class constructor.
3. this keyword can be used to invoke current class method (implicitly)
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this keyword can also be used to return the current class instance.


state
this
behaviour
reference
variable
object

The this keyword can be used to refer current class instance variable

- If there is ambiguity between the instance variable and parameter, this keyword resolves the problem of ambiguity.

```java
class Student10{
    int id;
    String name;

    Student10(int id,String name){
    id = id;
    name = name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student10 s1 = new Student10(18,"Virat");
    Student10 s2 = new Student10(3,"Suresh");
    s1.display();
    s2.display();
    }
}
```

Terminal

```
sh-4.3$ javac Student10.java
sh-4.3$ java Student10
0 null
0 null
sh-4.3$
```

- In the above example, parameter (formal arguments) and instance variables are same that is why we are using this keyword to distinguish between local variable and instance variable.

```java
class Student11{
    int id;
    String name;

    Student11(int id,String name){
    this.id = id;
    this.name = name;
    }
    void display(){System.out.println(id+" "+name);}
    public static void main(String args[]){
    Student11 s1 = new Student11(18,"Virat");
    Student11 s2 = new Student11(3,"Suresh");
    s1.display();
    s2.display();
}
}
```

Terminal

```
sh-4.3$ javac Student11.java
sh-4.3$ java Student11
18 Virat
3 Suresh
sh-4.3$
```

this() can be used to invoke current class constructor.

```java
class Student13{
    int id;
    String name;
    Student13(){System.out.println("default constructor is invoked");}

    Student13(int id,String name){
    this ();//it is used to invoke current class constructor.
    this.id = id;
    this.name = name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student13 e1 = new Student13(18,"Virat");
    Student13 e2 = new Student13(3,"Suresh");
    e1.display();
    e2.display();
   }
}
```

Terminal

```
sh-4.3$ javac Student13.java
sh-4.3$ java Student13
default constructor is invoked
default constructor is invoked
18 Virat
3 Suresh
sh-4.3$
```

Why to use this() constructor call

```java
class Student14{
    int id;
    String name;
    String city;

    Student14(int id,String name){
    this.id = id;
    this.name = name;
    }
    Student14(int id,String name,String city){
    this(id,name);//now no need to initialize id and name
    this.city=city;
    }
    void display(){System.out.println(id+" "+name+" "+city);}

    public static void main(String args[]){
    Student14 e1 = new Student14(18,"Virat");
    Student14 e2 = new Student14(3,"Suresh","Muradnagar");
    e1.display();
    e2.display();
  }
}
```

Terminal

```
sh-4.3$ javac Student14.java
sh-4.3$ java Student14
18 Virat null
3 Suresh Muradnagar
sh-4.3$
```