



# Object-Oriented Programming (CS F213)

**Module I: Object-Oriented and Java Basics**

**CS F213 RL 4.1: Accessing Object Fields and Methods**

**BITS Pilani**

**Dr. Pankaj Vyas**

Department of Computer Science, BITS-Pilani, Pilani Campus

# CS F213 RL 4.1 : Topics



- Accessing Class Members
  - ❑ Accessing Instance Fields of a Class
  - ❑ Invoking Methods of a Class
  - ❑ Implicit vs Explicit Parameters
  - ❑ Use of 'this' reference pointer

# What is the Problem?



- Problem

```
class XYZ
{
    private int x;
    private double y;

    // Constructor Method
    XYZ(int a, double b)
    {
        x=a; y = b;
    } // End of Constructor

    // Method to Read the Value of x
    public int getX()
    {
        return x;
    } // End of Method

    // Method to get the value of y
    public int getY()
    {
        return y;
    }
} // End of class XYZ

// Driver Class
class Test
{
    public static void main(String args[])
    {
        // How to Access the Instance Fields and Methods of XYZ Here
    } // End of Method
} // end of class Test
```

Instance Fields

Constructor Method

What is the Access Mechanism

# Accessing Object Methods and Fields of a Class



- Within (Inside the class) the class every instance field (even if it is private) and every object method is directly accessible.
- Outside the class only those Instance Fields and Object methods are accessible which are not private.
- Outside the class the Instance Fields and Object methods are accessed using object-references of that class
- Syntax for Accessing Non-private Instance Fields Outside the class

**<object-reference-name>.<instance-field-name>**

- Syntax for Accessing Non-private Object Methods Outside the class

**<object-reference-name>.object-method-name();**

**<object-reference-name>.object-method-name(parameters);**

# Example 1



// File Name : Demo.java

class XYZ

{

private int x;

private double y;

// Constructor Method

XYZ(int a, double b)

{

x=a;

y = b;

}// End of Constructor

// Method to Read the Value of x

public int getX()

{

return x;

} // End of Method

// Method to get the value of y

public int getY()

{

return y;

} // End of Method

}// End of class XYZ

// Driver Class

class Test

{

public static void main(String args[])

{

**// How to Access Instance Fields and  
// Methods of XYZ class in Test class ?**

**// Step 1 : Create an Instance of class XYZ**

**XYZ x1 = new XYZ(10, 10);**

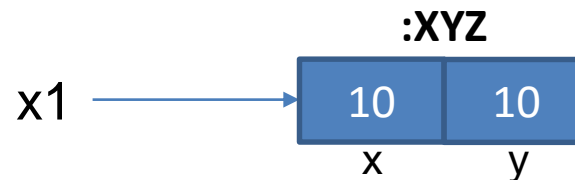
**// Step 2: Access via instance reference**

**int a = x1.getX();**

**int b = x1.getY();**

} // End of Method

}// end of class Test



getX() and getY()  
Methods are  
accessed  
via object-reference x1

# Example 2



// File Name : Demo.java

class XYZ

{

```
private    int x;    // private instance field
           int y;    // package-private instance field
protected int z;    // protected instance field
public     int c;    // public instance field
```

// Constructor Method

XYZ(int A, int B, int C, int D)

{

x = A;

y = B;

z = C;

c = D;

// End of Constructor

// Method to getSum of instance fields

public int getSum()

{

return x + y + z + c;

// End of Method

// Method displays the instance field values

public void display()

{

System.out.println("x = " + x);

System.out.println("y = " + y);

System.out.println("z = " + z);

System.out.println("c = " + c);

// End of Method

// End of class XYZ

// Driver class Test

class Test

{

public static void main(String[] args)

{

XYZ x1 = new XYZ(6, 7, 10, 23);

// x1.x = 13;

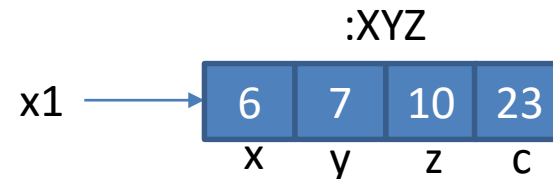
x1.y = 20;

x1.z = 40;

x1.c = 60;

// End of Method

// End of class Test



**Erroneous Statement: as 'x'  
is private in class XYZ**

# Example 2



// File Name : Demo.java

class XYZ

{

```
private    int x;    // private instance field
           int y;    // package-private instance field
protected int z;    // protected instance field
public     int c;    // public instance field
```

// Constructor Method

XYZ(int A, int B, int C, int D)

{

    x = A;

    y = B;

    z = C;

    c = D;

// End of Constructor

// Method to getSum of instance fields

public int getSum()

{

    return x + y + z + c;

// End of Method

// Method displays the instance field values

public void display()

{

    System.out.println("x = " + x);

    System.out.println("y = " + y);

    System.out.println("z = " + z);

    System.out.println("c = " + c);

// End of Method

// End of class XYZ

// Driver class Test

class Test

{

public static void main(String[] args)

{

XYZ x1 = new XYZ(6, 7, 10, 23);

// x1.x = 13;

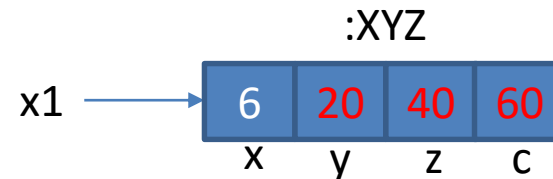
x1.y = 20;

x1.z = 40;

x1.c = 60;

// End of Method

// End of class Test



**Erroneous Statement: as 'x'  
is private in class XYZ**

# Example 2



// File Name : Demo.java

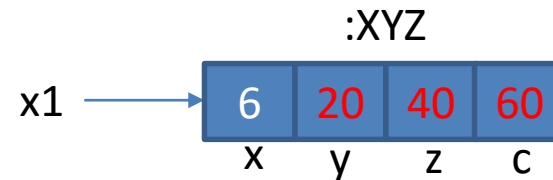
class XYZ

```
{  
    private    int x;        // private instance field  
    int y;        // package-private instance field  
    protected int z;        // protected instance field  
    public    int c;        // public instance field  
    // Constructor Method  
    XYZ(int A, int B, int C, int D)  
    {  
        x = A;  
        y = B;  
        z = C;  
        c = D;  
    }  
    // End of Constructor  
    // Method to getSum of instance fields  
    public int    getSum()  
    {  
        return x + y + z + c;  
    }  
    // End of Method  
    // Method displays the instance field values  
    public void    display()  
    {  
        System.out.println("x = " + x);  
        System.out.println("y = " + y);  
        System.out.println("z = " + z);  
        System.out.println("c = " + c);  
    }  
    // End of Method  
} // End of class XYZ
```

// Driver class Test

class Test

```
{  
    public static void main(String[] args)  
    {  
        XYZ x1 = new XYZ(6, 7, 10, 23);  
        // x1.x = 13;  
        x1.y = 20;  
        x1.z = 40;  
        x1.c = 60;  
        x1.display();  
    }  
} // End of Method  
} // End of class Test
```



F:\>java Test

x = 6

y = 20

z = 40

c = 60

OUTPUT



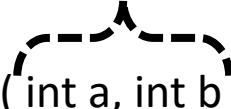
# Implicit vs Explicit Method Parameters



- Any object in memory referenced by a object-reference-variable 'r' is the implicit parameter to any method which is invoked via 'r'
- Explicit parameters are the part of method signature
- Example

```
class XYZ
{
    .....
    public void
    {
        .....
    } // End of Method
} // End of class XYZ
```

**Explicit Parameters for doS()**

doS( int a, int b )

**// Driver Code**

**XYZ x1 = new XYZ();**

**x1.doS(4,6);**

**'x1' is implicit parameter**

# Implicit Parameter Examples

```
// File Name : Demo.java
class AB
{
    private int a, b; // Instance Fields
    // Constructor Method
    AB(int x, int y)
    {
        a = x; b = y;
    } // End of Method
    // Method to display instance field values
    public void display()
    {
        System.out.println(" a= " + a);
        System.out.println(" b = " + b);
    } // End of Method
} // End of class AB
```

```
// Driver Code
class Test
{
    public static void main(String args[])
    {
        AB    a1 =    new AB( 4 , 8);
        AB    a2 =    new AB( 3 , 5);
        AB    a3 =    new AB( 7 , 21);
        AB    a4 =    new AB( 2 , 9);

        a1.display();
        a2.display();
        a3.display();
        a4.display();

    } // End of Method
} // End of class Test
```

**states of the objects referenced by variables 'a1' .. 'a4' are the default parameters for the display() method**

# Use of this pointer

- 'this' is a Java Keyword which always points to the object invoking the method (i.e. implicit parameter )
- 'this' pointer helps to differentiate the local-variables from instance fields especially when they have same name
- Example

```
class AB  
{
```

```
    private int a;  
    private int b;  
    // Constructor Method
```

```
    AB(int a, int b)  
    {
```

```
        this.a = a;  
        this.b = b;
```

```
    } // End of Method
```

```
} // End of class AB
```

Instance Fields  
and constructor  
Arguments have  
Same name

this.a refers to  
instance-field 'a'

this.b refers to  
instance-field 'b'

# this pointer Example



```
// File Name : Demo.java
```

```
class AB
```

```
{
```

```
    private int a;
```

```
    private int b; // Instance Fields
```

```
    // Constructor Method
```

```
    AB(int x, int y)
```

```
    {
```

```
        this.a = x;
```

```
        this.b = y;
```

```
        this.display();
```

// you can write simple 'display()' also

```
    } // End of Method
```

```
    // Method to display instance field values
```

```
    public void display()
```

```
    {
```

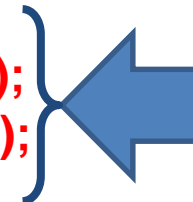
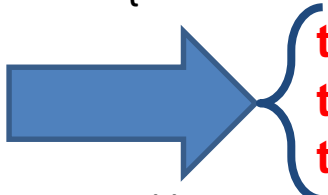
```
        System.out.println(" a= "+ this.a);
```

```
        System.out.println(" b = " +this.b);
```

```
    } // End of Method
```

```
} // End of class AB
```

Refer to  
Instance-fields  
via 'this'



Refer to  
Instance-fields  
via 'this'

---

***Thank You***