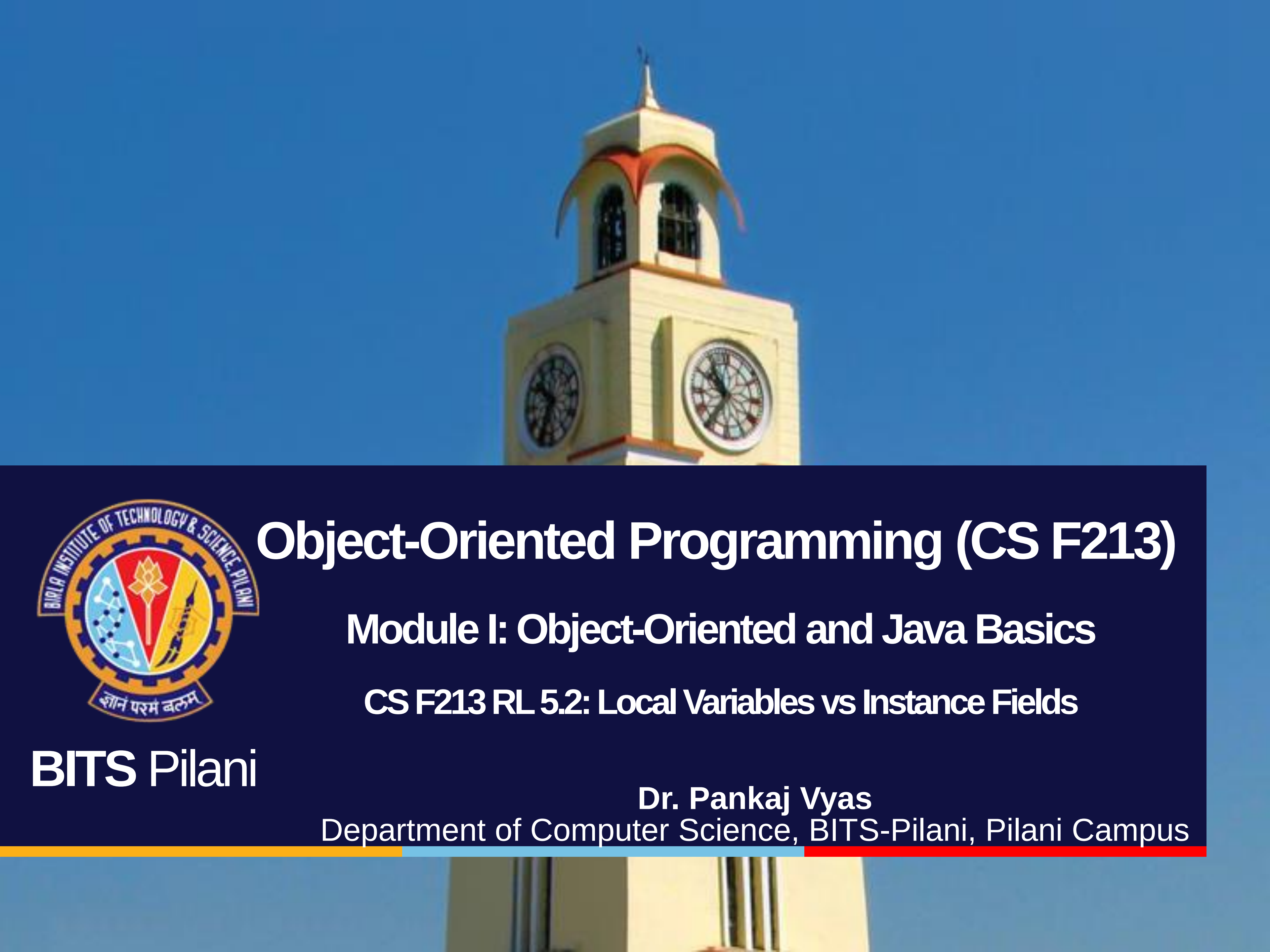


Object-Oriented Programming (CS F213)

Module I: Object-Oriented and Java Basics

CS F213 RL 5.2: Local Variables vs Instance Fields

BITS Pilani

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 5.2 : Topics

- Local Variable vs Instance Fields

Local Variables

```
class  XYZ
{
    private      int      a;
    public       static   int      b;

    public       void     doS(int  m, int n)
    {
            int y;
    }
}
```

Instance Field

Class/static Field

Local Arguments

Local Variable

- **Instance-Field ‘a’ is visible only inside class XYZ**
- **Class Field ‘b’ is visible everywhere**
- **Local arguments ‘m’ and ‘n’ and variable ‘y’ of doS() method are accessible only in side doS().**

Local Variables vs Object Variables/Instance Fields

Instance Fields / Object Variables	Local Variables

Local Variables vs Object Variables/Instance Fields

Instance Fields / Object Variables	Local Variables
1. Can have Access Modifiers : public, private , ….	1. Local Variables have only local or a block scope. So, Access Modifiers are not allowed.

Local Variables vs Object Variables/Instance Fields

Instance Fields / Object Variables	Local Variables
1. Can have Access Modifiers : public, private , ….	1. Local Variables have only local or a block scope. Access Modifiers not allowed
2. <static> keyword can be applied for class variables	2. <static< keyword can not be used for local variables

Local Variables vs Object Variables/Instance Fields

Instance Fields / Object Variables	Local Variables
1. Can have Access Modifiers : public, private , ….	1. Local Variables have only local or a block scope. Access Modifiers not allowed
2. <static> keyword can be applied for class variables	2. <static< keyword can not be used for local variables
3. Every instance field is auto initialized upon object creation. [int, short, long and byte types are auto initialized to 0s, float and double types are auto initialized to 0.0f and 0.0 values respectively, boolean type variable is auto initialized to ‘false’ value]	3. Local variables are not auto initialized. They have to be explicitly initialized to some default value before their use.

Local Variables Example 1

No Access Modifier and No static keyword For Local Variables

```
// File Name : Test.java
class Test
{
    public      static  void   main(String args[])
    {
        public int      a;
        static float    b;
    } // End of Method
} // End of Class Test
```

Compile-Time Error
Local Variables cannot have Access Modifiers : public, private etc

Compile-Time Error
'static' keyword cannot be used with local variables

Local Variables Example 2

- Local Variables have to be Explicitly Initialized.
- No Default Initialization

```
// File Name : Test.java
class Test
{
    public      static  void   main(String args[])
    {
        int     a;
        float   b = a + 10;
    } // End of Method
} // End of Class Test
```

Compile-Time Error
Local variables are to be initialized explicitly to some default value before their use

Local Variables Example 3

- Local Variables can have only ‘final’ declarations. Note ‘final’ local variables means it can not change its values

```
// File Name : Test.java
class Test
{
    public      static  void    main(String args[])
    {
        final   int     a = 56;
        float   b = a + 10;
    } // End of Method
} // End of Class Test
```

final local Variable

final local variables cannot change their values

Auto Initialization of Instance Fields

- Every Instance-Field is auto-initialized to some default value (even if no value is assigned) according its type as per following table

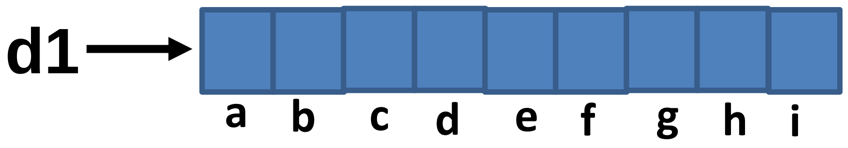
Type of Instance-Field	Default Value
byte	0
short	0
int	0
long	0
char	‘’
float	0.0
double	0.0
boolean	false
Any class type	null

Auto Initialization of Instance Fields : Example

```
// File Name: Demo.java
class A
{
} // End of class A
class Demo
{
  private byte                  a;
  private short                 b;
  private int                   c;
  private long                  d;
  private char                  e;
  private float                 f;
  private double                g;
  private booelan               h;
  private A                     i;
  // Method to display the Values of Instance Fields
   public void display()
   {
    System.out.println("a=" +this.a);
    System.out.println("b=" +this.b);
    System.out.println("c=" +this.c);
    System.out.println("d=" +this.d);
    System.out.println("e=" +this.e);
    System.out.println("f=" +this.f);
    System.out.println("g=" +this.g);
    System.out.println("h=" +this.h);
     System.out.println("h=" +this.i);
   } // End of Method
}// End of class Demo
```

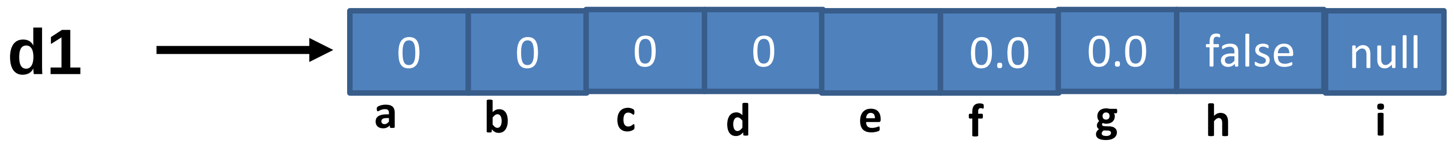
```
// Driver Class
class Test
{
    // Driver Method
    public static void main(String args[])
    {
            Demo  d1  =  new  Demo();

            d1.display();

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

What values are assigned here?



Auto Initialization of Instance Fields : Example

Values are assigned as Follows?



```
F:\>javac Demo.java

F:\>java Test
a=0
b=0
c=0
d=0
e=
f=0.0
g=0.0
h=false
i=null
```

Thank You