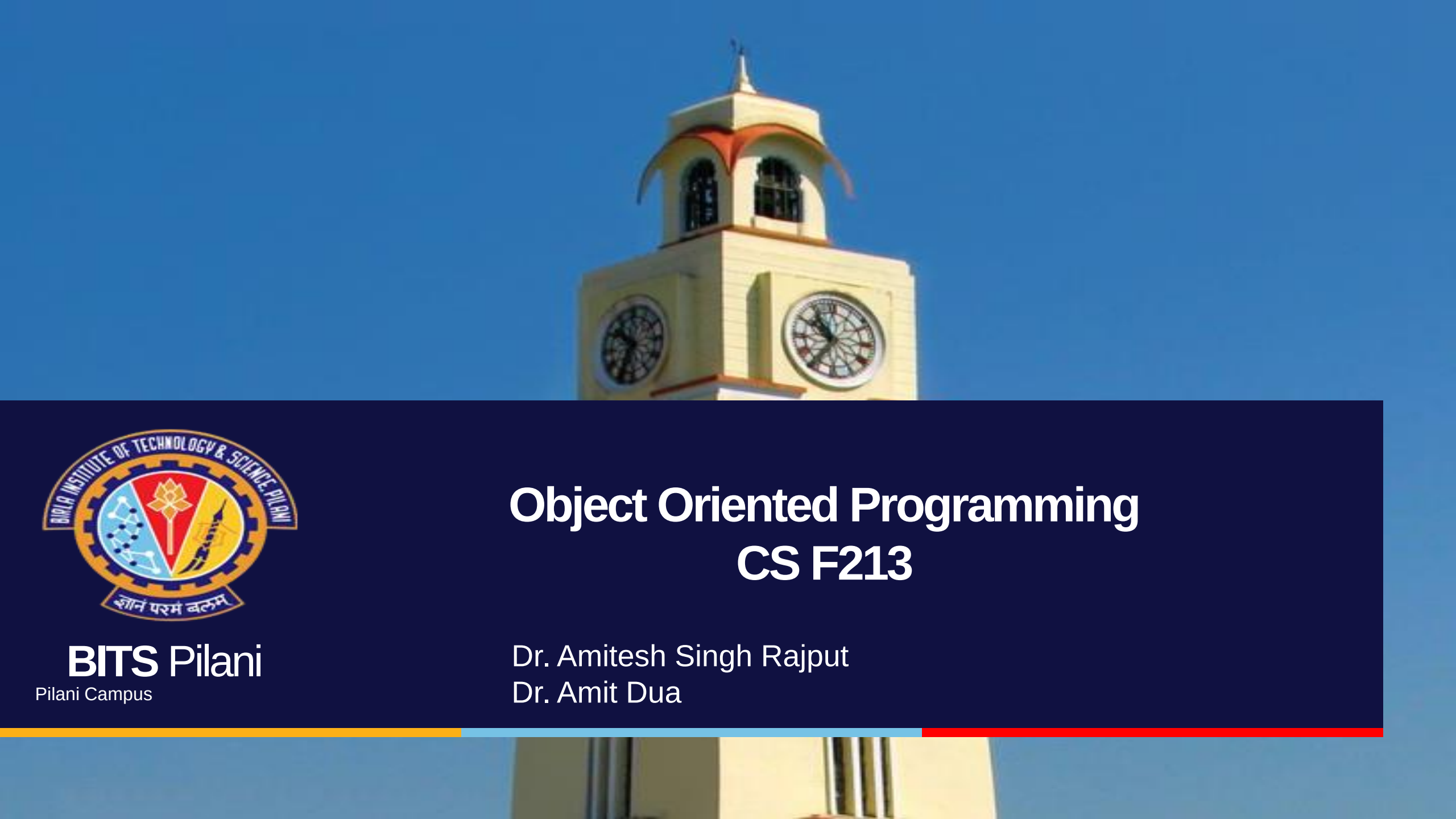

Object Oriented Programming
CS F213

BITS Pilani
Pilani Campus

Dr. Amitesh Singh Rajput
Dr. Amit Dua

Inheritance – private vs. final methods

```
public class ParentOne {
  int a, b;
  private void showOne() {
   System.out.println("a = "+a+" b = "+b);
  }
}
```

Private methods are not accessible from outside of the class.

```
public class ParentOne {
  int a, b;
  final void showOne() {
   System.out.println("a = "+a+" b = "+b);
  }
}
```

final methods can not be overridden by the child class.

```
public class ChildOne extends ParentOne{
  public static void main(String args[]){
        ChildOne c1 = new ChildOne();
        c1.showOne();
  }
}
```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The method showOne() from the type ParentOne_Private is not visible

Arrays

BITS Pilani

Pilani Campus

Arrays

- Collection of similar datatypes that shares common name.
- Each datatype in an array is called "element“.
- Counting of element is called "index" or "subscript“.
- Index always starts from 0.
- Total number of elements is referred as "length" or "size" of array.

Arrays

- Syntax to declare an array
  - `int[] arr;`
  - `int []arr;`
  - `int arr[];`

It tells the compiler that this variable (arr) will hold an array of the integer type.

- Initialization of an array
  - `arr = new int[size];`

It links arr with an actual, physical array of integers

- Arrays can be accessed using
  - **Simple for loop**
  - **For each loop**
  - **Labelled for loop**

Arrays – Practice: Normal Prog.


```
class Abc{

    public static void main (String[] args){

       int[] arr;
       arr = new int[5];

       arr[0] = 10;
       arr[1] = 20;
       arr[2] = 30;
       arr[3] = 40;
       arr[4] = 50;

       for (int i = 0; i < arr.length; i++)
          System.out.println("Element at index " + i + " : "+ arr[i]);
    }
}
```

Arrays – Practice: Class Prog.


```
class Employee{
    public int emp_no;
    public String name;
    Employee(int emp_no, String name){
        this.emp_no = emp_no;
        this.name = name;
    }
}
```

```
public class Abc{
    public static void main (String[] args) {
        Employee[] arr;
        arr = new Employee[5];

        arr[0] = new Employee(1,"Ankit");
        arr[1] = new Employee(2,"Vaibhav");
        arr[2] = new Employee(3,"Sahni");
        arr[3] = new Employee(4,"Sachin");
        arr[4] = new Employee(5,"Rahul");

        for (int i = 0; i < arr.length; i++)
          System.out.println("Element at " +i+" :
          "+arr[i].emp_no +" "+arr[i].name);
    }
}
```

Output:

Element at 0 : 1 Ankit
Element at 1 : 2 Vaibhav
Element at 2 : 3 Sahni
Element at 3 : 4 Sachin
Element at 4 : 5 Rahul

For each loop

Starts with the keyword **for** like a normal for-loop.

Declare a variable that is of the same type as of the array, followed by a colon, followed by the array name.

In loop body, use the created loop (no indexed array element).

```
int arr[]={12,23,44,56,78};

  for(int i:arr){
      System.out.println(i);   //Printing array using for-each loop
  }
```

Labelled For Loop

A valid variable name that represents the loop name to where the control of execution should jump.

```
aa:
 for(int i=1;i<=3;i++){
        bb:
         for(int j=1;j<=3;j++){
             if(i==2 && j==2){
                 break bb;
             }
             System.out.println(i+" "+j);
         }
 }
```

Copying a Java Array

- arraycopy method of the System class is used to copy an array to another.

public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)

```
int a[]= {2,3,5};
int b[] = new int[a.length];

System.arraycopy(a, 1, b, 0, a.length-1);

for(int i=0; i<b.length; i++)
    System.out.print(" "+b[i]);
```

Output:

3 5 0

Printing an Array – Alternative approach


```
public class Test{
 public static void main(String args[]){
    int arr[] = {1, 2, 3};
    System.out.println(arr);
    System.out.println(Arrays.toString(arr));
 }
}
```

Output:

[I@2a139a55
[1, 2, 3]

Array Class (import java.util.*)

static type	**binarySearch**(type[] a, type key) Searches the specified array of type for the specified value using the binary search algorithm.
static boolean	**equals**(type[] a, type[] a2) Returns true if the two specified arrays of type are equal to one another.
static void	**fill**(type[] a, type val) Assigns the specified type value to each element of the specified array of type.
static void	**fill**(type[] a, int fromIndex, int toIndex, type val) Assigns the specified type value to each element of the specified range of the specified array of types.
static void	**sort**(type[] a) Sorts the specified array of type into ascending numerical order.
static void	**sort**(type[] a, int fromIndex, int toIndex) Sorts the specified range of the specified array of type into ascending numerical order.
	type = byte, char, double, float, int, long, short, Object

Array Class - Example

```
int a[]= {2,3,5,1,4,7};

for(int i=0; i<a.length; i++)
System.out.println(a[i]+" ");

Arrays.sort(a, 0, 4);
System.out.println(Arrays.toString(a));

Arrays.sort(a);
System.out.println(Arrays.toString(a));

System.out.println("Binary Search for 5 is “ + Arrays.binarySearch(a,5));
```

Output:

2 3 5 1 4 7
[1, 2, 3, 5, 4, 7]
[1, 2, 3, 4, 5, 7]
Binary Search for 5 is 4

Array Class - Example


```
int a[]= {1,2,3,4,5,7};

System.out.println(Arrays.toString(Arrays.copyOf(a, a.length)));

System.out.println(Arrays.toString(Arrays.copyOfRange(a, 1, 4)));

Arrays.fill(a, 4, a.length, 1);
System.out.println(Arrays.toString(a));

Arrays.fill(a, 1);
System.out.println(Arrays.toString(a));
```

Output:
[1, 2, 3, 4, 5, 7]
[2, 3, 4]
[1, 2, 3, 4, 1, 1]
[1, 1, 1, 1, 1, 1]

Predict the output

```
int arr1[] = {1, 2, 3};
int arr2[] = {1, 2, 3};
if (arr1 == arr2)
   System.out.println("Same");
else
   System.out.println("Not same");
```

Output: Not same

Problem: == compares the array references

Solution: Arrays.equals(arr1, arr2)

Predict the output


```
int inarr1[] = {1, 2, 3};
int inarr2[] = {1, 2, 3};

Object[] arr1 = {inarr1, inarr2};
Object[] arr2 = {inarr2, inarr1};

if (Arrays.equals(arr1, arr2))
   System.out.println("Same");
else
   System.out.println("Not same");
```

Output: Not same

Solution: Arrays.deepEquals(arr1, arr2)

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You!