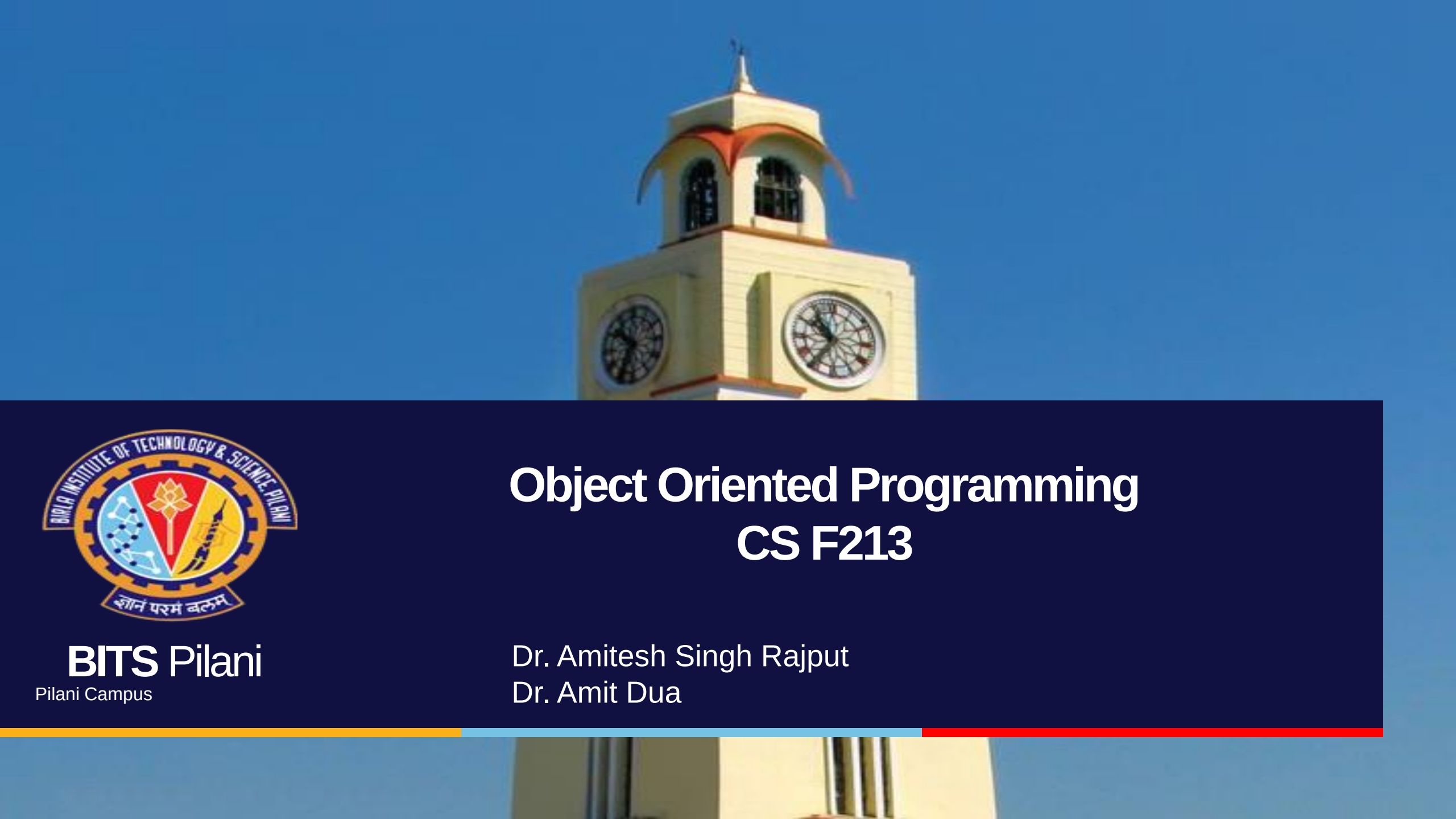


Object Oriented Programming
CS F213

BITS Pilani
Pilani Campus

Dr. Amitesh Singh Rajput
Dr. Amit Dua

‘this’ Keyword

‘this’ Keyword

- It is a reference variable that refers to the current object.

- Six usage

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly).
3. this can be used to invoke current class constructor.
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 can be used to return the current class instance from the method.

this: to refer current class instance variable


```
class Account{
  int acc;
  String name;
  float amount;
  Account(int acc, String name, float amount){
   acc = acc;
   name = name;
   amount = amount;
  }
  void display(){
   System.out.println(acc+" "+name+" "+amount);
  }
}
class TestAccount{
  public static void main(String[] args){
   Account a1 = new Account(832345,"Ankit",5000);
   a1.display();
  }
}
```

Output:
0 null 0.0

this: to refer current class instance variable


```
class Account{
  int acc;
  String name;
  float amount;
  Account(int acc, String name, float amount){
   this.acc = acc;
   this.name = name;
   this.amount = amount;
  }
  void display(){
   System.out.println(acc+" "+name+" "+amount);
  }
}
class TestAccount{
  public static void main(String[] args){
   Account a1 = new Account(832345,"Ankit",5000);
   a1.display();
  }
}
```

Output:
832345 Ankit 5000

this: to invoke current class method

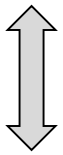

```
class Account{
  int acc;
  String name;
  float amount;
  void insert(int acc,String name, float amount){
        this.acc = acc;
        this.name = name;
        this.amount = amount;
        this.display();
  }
  void display(){
        System.out.println(acc+" "+name+" "+amount);}
  }
class TestAccount{
  public static void main(String[] args){
        Account a1 = new Account();
        a1.insert(832345,"Ankit",5000);
  }
}
```

What if we remove “this” from here?

Output:
832345 Ankit 5000

this() : To invoke current class constructor

- The this() constructor call can be used to invoke the current class constructor.
- In other words, it is used for **constructor chaining**.
- Calling default constructor from parameterized constructor.



- Calling parameterized constructor from default constructor.

Constructor Chaining
Example


```
class Account{
  int acc;
  String name;
  float amount;
  Account(int acc, String name){
   this.acc = acc;
   this.name = name;
  }
  Account(int acc, String name, float amount){
   this.acc = acc;
   this.name = name;
   this.amount = amount;
  }
  void display(){
   System.out.println(acc+" "+name+" "+amount);
  }
}
```

```
class TestAccount{
  public static void main(String[] args){
   Account a1 = new Account(832345,"Ankit",5000);
   a1.display();
  }
}
```

```
this(acc, name);  //reusing constructor
```

‘this’ Keyword

- It is a reference variable that refers to the current object

- Six usage

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly).
3. this can be used to invoke current class constructor.
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 can be used to return the current class instance from the method.

this: to pass as an argument in the method


```
class Account{
  int acc;
  String name;
  float amount;
  Account(int acc, String name){
    this.acc = acc;
    this.name = name;
    display(this);
  }
  void update(int act, String aname, float amt){
    acc = act;
    name = aname;
    amount = amt;
    display(this);
  }
  void display(Account a){
    System.out.println(a.acc+" "+a.name+" "+a.amount);}
}
```

```
class Second{
 public static void main(String[] args){
   Account a1 = new Account(832345,"Ankit");
   Account a2 = new Account(832345,"Shobit");
   a1.update(832346, "Sahni", 5000);
 }
}
```

Output:

832345 Ankit 0.0
832345 Shobit 0.0
832346 Sahni 5000.0

this: to pass as argument in the constructor call

```
class Account{
  int acc;
  String name;

Account(int acc, String name){
          this.acc = acc;
          this.name = name;
          Branch b = new Branch(this);
          b.display();
  }
}
```

```
class Branch{
  Account obj;
  int branch;
  Branch(Account obj){
          this.obj = obj;
          this.branch = 111;
  }
  void display(){
    System.out.println(obj.acc + " " + obj.name
   + " " + branch);
  }
}
class TestAccount{
  public static void main(String args[]){
   Account a1 = new Account(832345,"Ankit");
  }
}
```

Output:
832345 Ankit 111

Returning Objects using this keyword


```
class Account{
 int acc;
 String name;
 float amount;
 Account(int acc, String name){
        this.acc = acc;
        this.name = name;
 }
 Account update(int act, String aname, float amt){
        acc = act;
        name = aname;
        amount = amt;
        return this;
 }
 void display(){
        System.out.println(acc+" "+name+" "+amount);}
}
```

```
class TestAccount{
  public static void main(String[] args){
    Account a1 = new Account(832345,"Ankit");
    a1.display();
    a1 = a1.update(832346, "Aankit", 5000);
    a1.display();
  }
}
```

innovate achieve lead

BITS Pilani

Pilani Campus

More on ‘Static’ Keyword

Overloading Static Method

```
public class Main{
  static int a ;
  static float b;
  static void assign(int A){
   a = A;
  }
  static void assign(int A, float B){
   a = A;
   b = B;
  }
  public static void main(String []args){
    Main.assign(10);
    System.out.println("Values are: a = "+a+" b = "+b);
    Main.assign(20,3.2f);
    System.out.println("Values are: a = "+a+" b = "+b);
  }}
```

Output

Values are: a = 10 b = 0.0
Values are: a = 20 b = 3.2

Static Block

- Used for initializing static variables.
- Static block is executed when the class is loaded in the memory.
- A class can have multiple static blocks that execute in the same sequence in which they are written in the program.

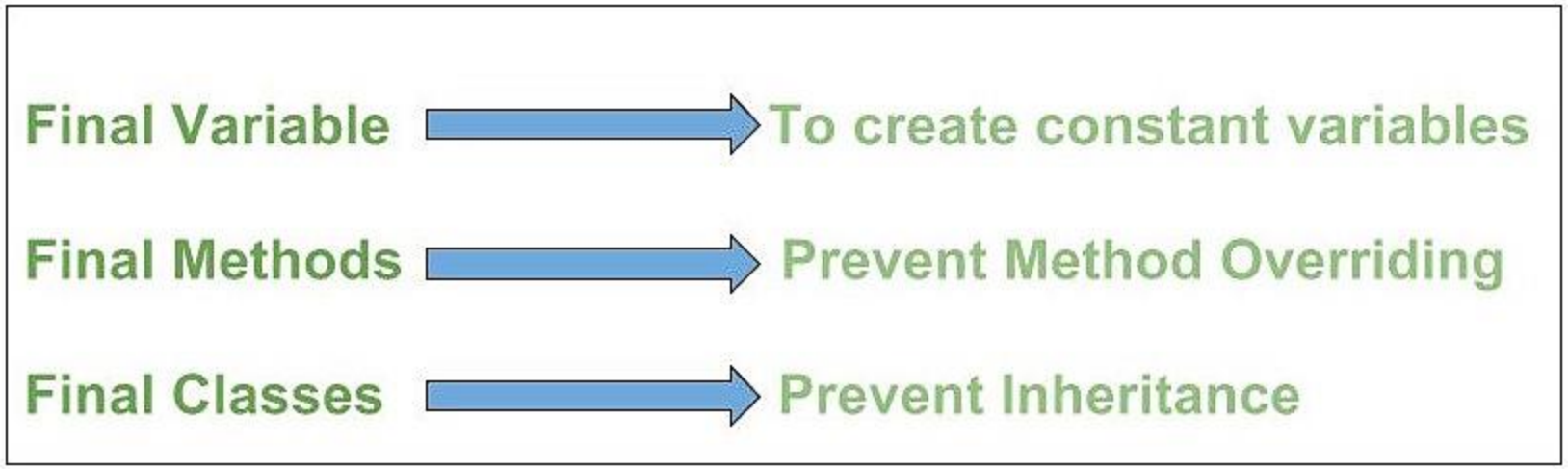
Static Block


```
class Abc{
  static int i;
  int j;

  static {          // static block
       i = 20;
       System.out.println("Inside static block");
  }
}

class Xyz {
  public static void main(String args[]) {
       System.out.println(Abc.i);
  }
}
```

Output

Inside static block
20

‘final’ Keyword



innovate achieve lead

BITS Pilani

Pilani Campus

Mutable and Immutable Objects

Mutable/Immutable Instances

- A mutable object can be changed after it is created.
- Vice-versa for immutable object.

- E.g. Immutable: java.lang.String
  Mutable: Object of the Account class (previous examples)

Mutable class Example


```
class Abc {
    public String str;
    Abc(String str) {
        this.str = str;
    }
    public String getName() {
        return str;
    }
    public void setName(String newname) {
        this.str = newname;
    }

    public static void main(String[] args) {
         Abc obj = new Abc("George");
         System.out.println(obj.getName());
         obj.setName("Federer");
         System.out.println(obj.getName());
    }
}
```

Output

```
George
Federer
```

How to create an Immutable class?

- Class must be declared as final
  - So that child classes can't be created
- Data members in the class must be declared as final
  - So that we can't change their value after object creation
- A parameterized constructor
- Getter method for all the variables in it
- No setters
  - To avoid any option to change the value of the instance variable

Immutable Class - Example

```
final class Account{
   final int acc;
   final String name;
   final float amount;
   Account(int acc, String name, float amt){
        this.acc = acc;
        this.name = name;
        this.amount = amt;
   }
   int getAcc(){
        return acc;
   }
   String getName() {
        return name;
   }
   float getAmount() {
        return amount;
   }}
```

Output:

```
Exception in thread "main" java.lang.Error:
Unresolved compilation problem:
The final field Account.amount cannot be
assigned
```

```
class TestAccount{
   public static void main(String[] args) {
   Account a = new Account(111,"Ankit",5000);
   System.out.println("Acc: "+a.getAcc()+" Name: "+a.name);
   a.amount = 1000;
   }
}
```

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You!