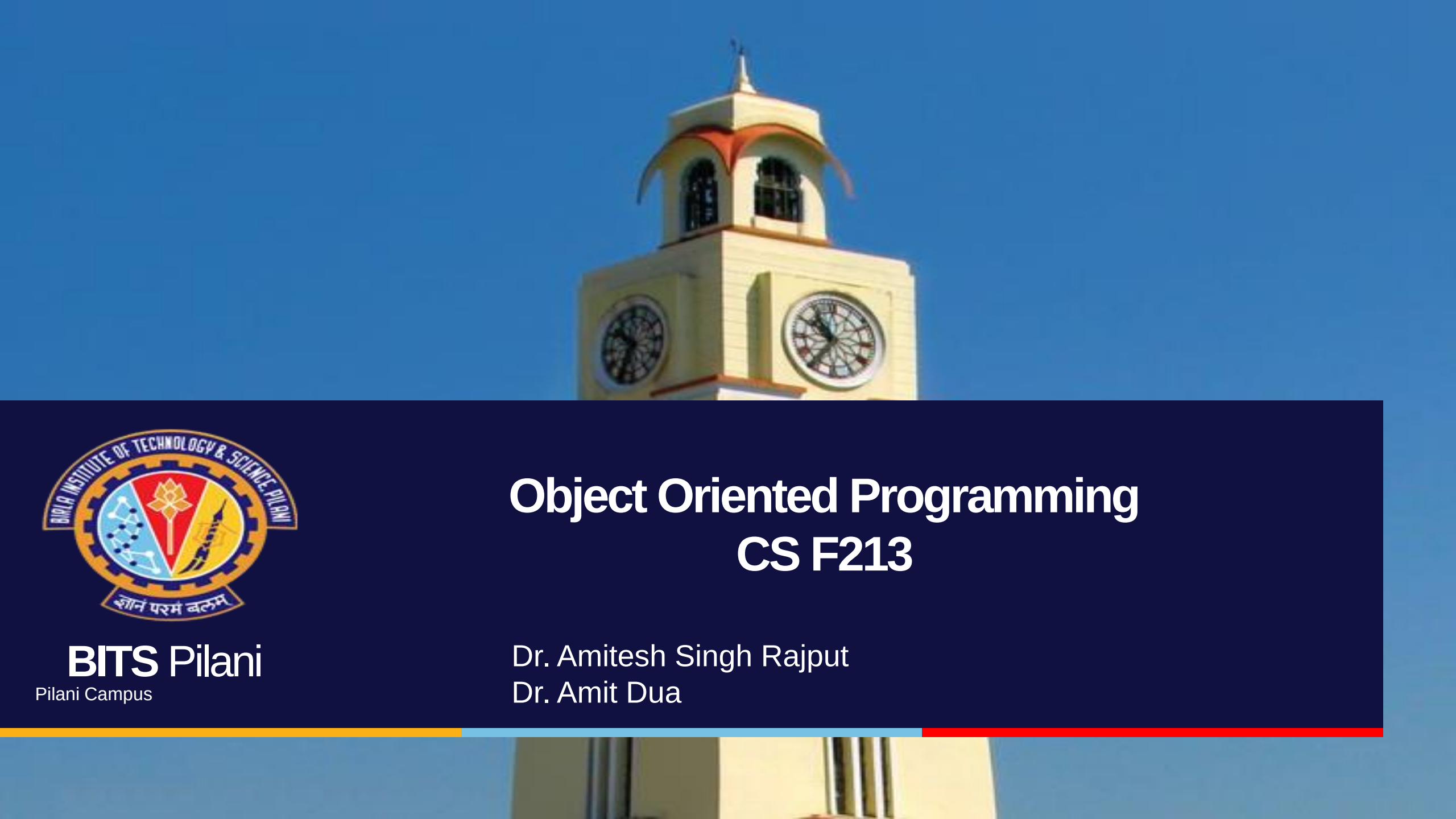
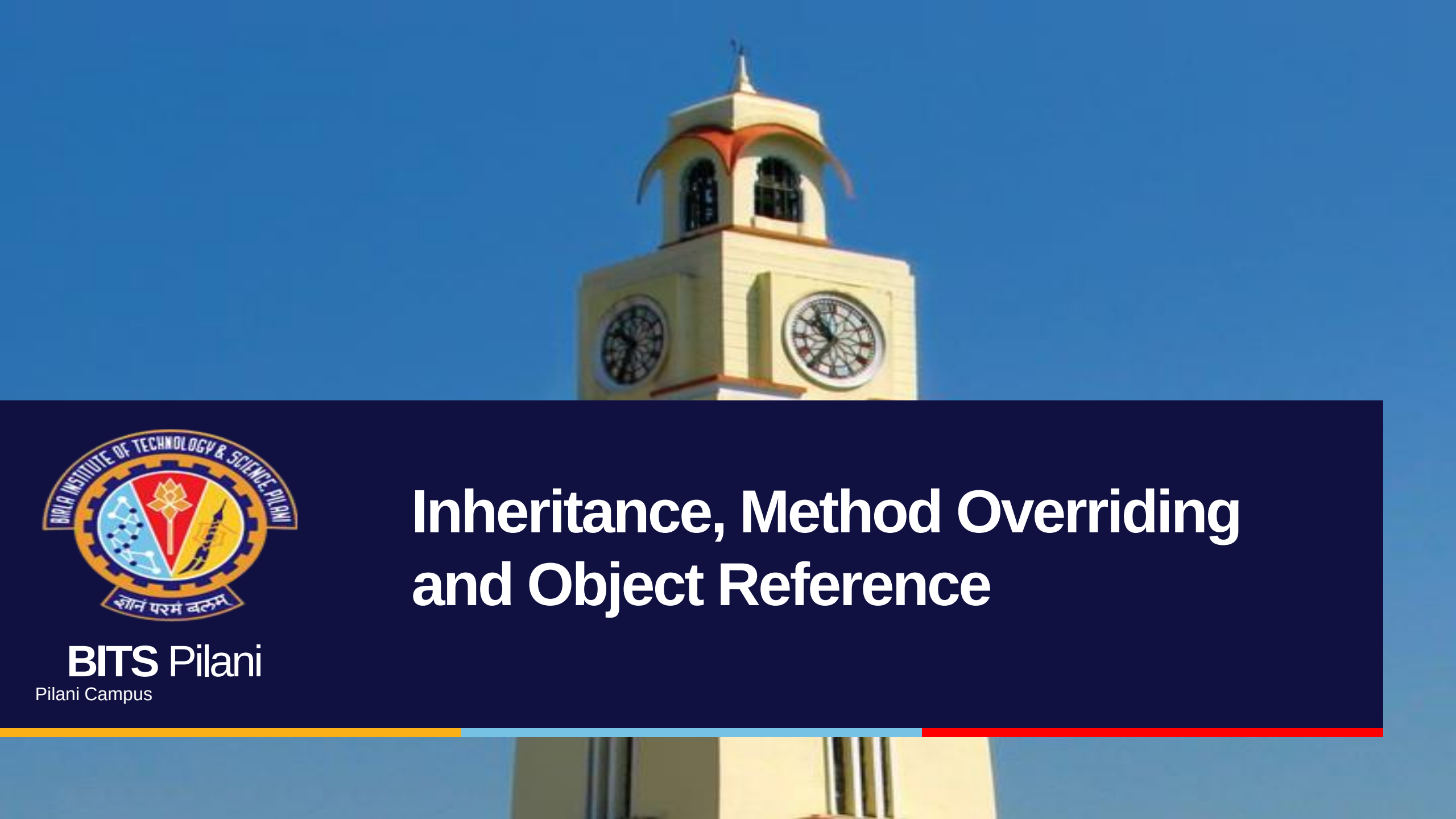


Object Oriented Programming
CS F213

BITS Pilani
Pilani Campus

Dr. Amitesh Singh Rajput
Dr. Amit Dua



Inheritance, Method Overriding and Object Reference

BITS Pilani
Pilani Campus

Recalling Inheritance

From view of the derived class there are 3 categories of methods in the base class.

1. Constructors:

To call constructors of base class "**super**" keyword is used

```
Syntax: super(<args>);
```

2. Overridden Functions:

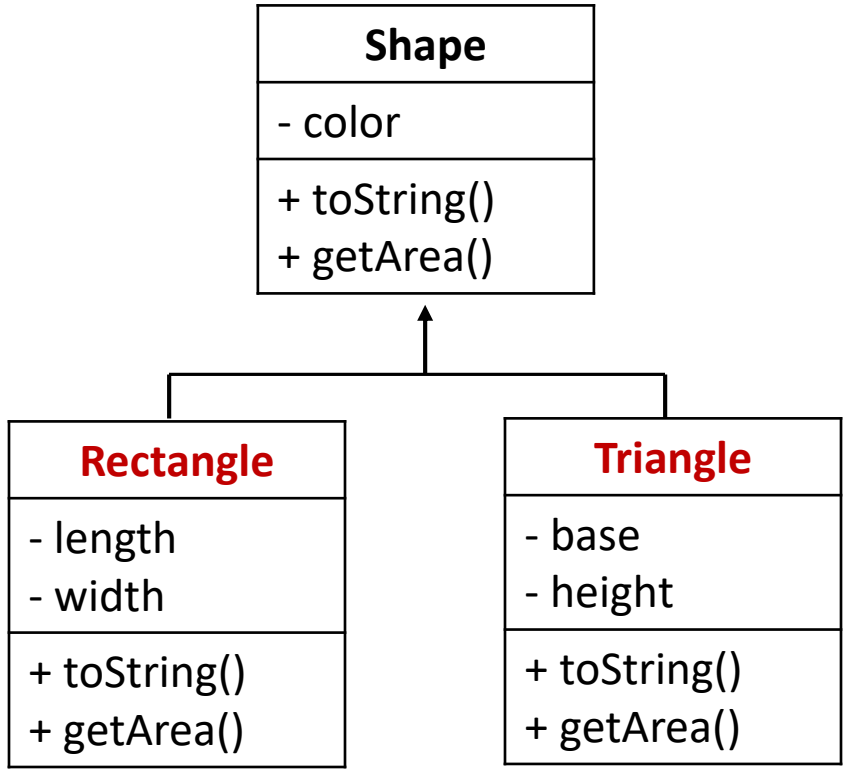
To call overridden functions of base class

```
Syntax: super.functionName(<args>);
```

3. Remaining/Normal Functions:

To call remaining functions

```
Syntax: functionName(<args>);
```



Recalling Inheritance - Example

```
public class Shape {
 String color;

 public Shape (String c) {
  color = c;
 }

 @Override
 public String toString() {
  return "Shape of color=\"" + color + "\"";
 }

 public double getArea() {
  System.err.println("Shape unknown! Cannot compute area!");
  return 0;
 }
}
```

Recalling Inheritance - Example

```
public class Rectangle extends Shape {
 int length, width;

 public Rectangle(String color, int l, int w) {
   super(color);
   length = l;
   width = w;
 }
 @Override
 public String toString() {
  return "Rectangle of length=" + length + " and width=" + width + ", subclass of " + super.toString();
 }
 @Override
 public double getArea() {
  return length*width;
 }
}
```

Recalling Inheritance - Example

```
public class Triangle extends Shape{
 int base, height;
 public Triangle(String color, int b, int h) {
   super(color);
   base = b;
   height = h;
 }

 @Override
 public String toString() {
  return "Triangle of base=" + base + " and height=" + height + ", subclass of " + super.toString();
 }

 @Override
 public double getArea() {
   return 0.5*base*height;
 }
}
```

Recalling Inheritance - Example

```
public class TestShape {
 public static void main(String[] args) {

 Rectangle s1 = new Rectangle("red", 4, 5);
 System.out.println(s1);
 System.out.println("Area is " + s1.getArea());

 Triangle s2 = new Triangle("blue", 7, 8);
 System.out.println(s2);
 System.out.println("Area is " + s2.getArea());
 }
}
```

```
Rectangle of length=4 and width=5, subclass of Shape of color="red"
Area is 20.0

Triangle of base=7 and height=8, subclass of Shape of color="blue"
Area is 28.0
```

Passing Objects to Methods

- Java is strictly pass by value.
- Call by reference can be achieved when objects are passed as arguments.
  - When a variable of class type is created, it implies that a reference to an object is created. E.g: `Account a1;`
  - Reference variable is used to store the address of the object.
  - When the reference is passed to a method, the parameter that receives refer to the same object.

Passing Objects - Example


```
class Account{
   int id; String name; float amount;
   Account(int act, String aname){
        id = act;
        name = aname;
   }
   boolean equalTo(Account a) {
        return(id == a.id && name == a.name);
   }
}
     class TestAccount{
       public static void main(String[] args){
            Account a1 = new Account(832345,"Ankit");
            Account a2 = new Account(832345,"Ankit");
            Account a3 = new Account(832346,"Shobit");
            System.out.println("a1==a2: " + a2.equalTo(a1));
            System.out.println("a1==a3: " + a3.equalTo(a1));
     }}
```

Output:

```
a1 == a2: true
a1 == a3: false
```

Assigning Object Reference Variables

- Value of a reference variable can be assigned to another reference variable.
- Assigning reference will not create distinct copies of objects.
- All reference variables are referring to the same object.

Assigning Object Reference

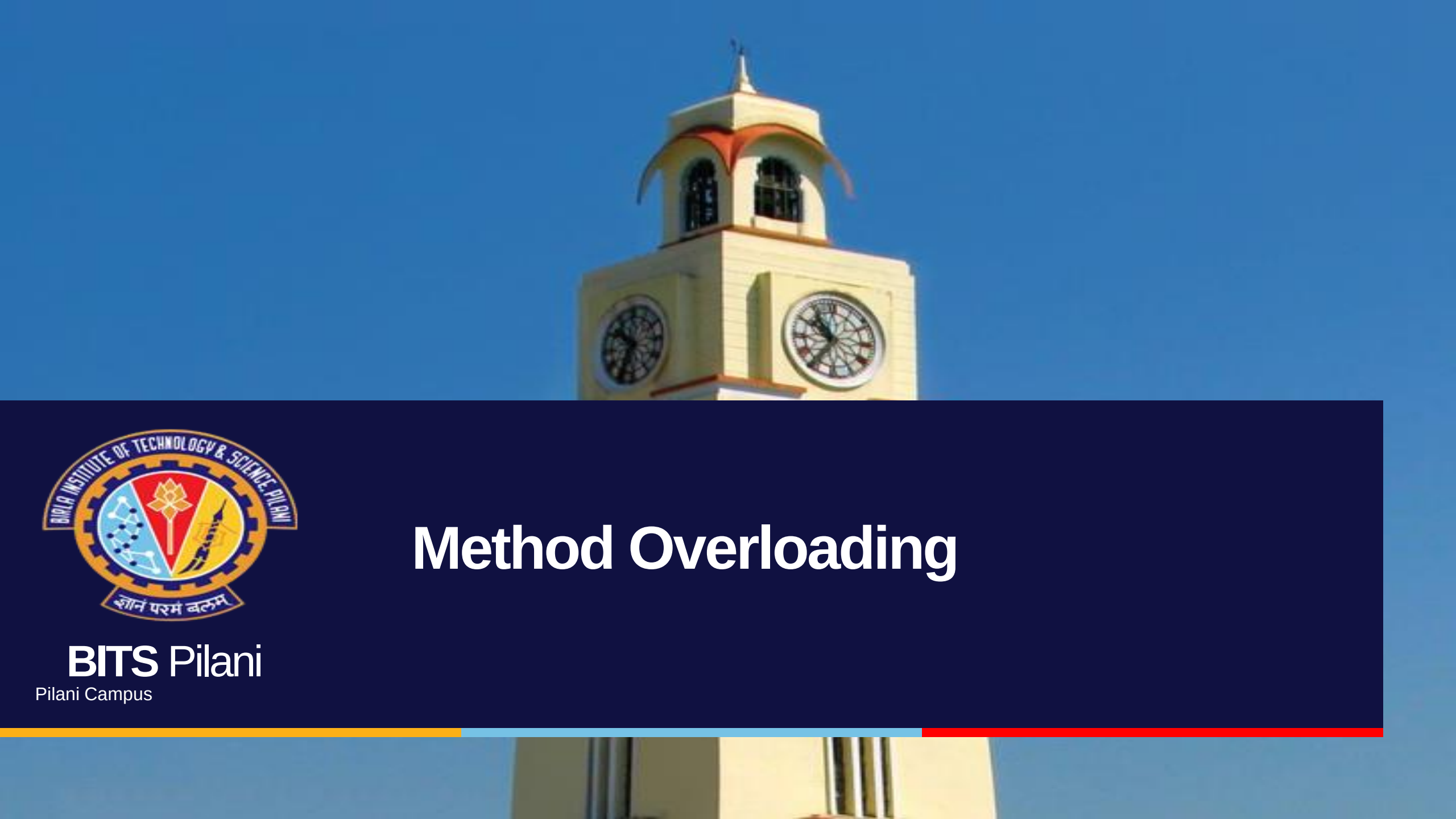

```
class Account{
   int acc;
   String name;
   float amount;
   Account(int act,String aname){
        acc = act;
        name = aname;
   }
   boolean equalTo(Account a) {
        return(acc == a.acc && name == a.name);
   }
   void display(){
        System.out.println(acc+" "+name+" "+amount);}
   }
```

Assigning Object Reference


```
class Second{
   public static void main(String[] args){
   Account a1 = new Account(832345,"Ankit");
   Account a2 = a1;
   Account a3 = new Account(832346,"Shobit");

   System.out.println("a1==a2:" + a2.equalTo(a1));
   System.out.println("a1==a3:" + a3.equalTo(a1));

   a1.name = "Sahni";
   a2.display();

   System.out.println(a1.hashCode());
   System.out.println(a2.hashCode());
   }
}
```

Output:

a1==a2: true
a1==a3: false
832345 Sahni 0.0



Method Overloading

BITS Pilani

Pilani Campus

Different ways to overload

- Changing the number of arguments
  - `int add(int a, int b)`
  - `int add(int a, int b, int c)`
- Changing the data type
  - `int add(int a, int b)`
  - `double add(double a, double b)`
- **Note:** Changing only return type does not mean method overloading
  - `int add(int a,int b)`
  - `double add(int a,int b)`
  - Compile Time Error: method add(int,int) is already defined in class Adder

Method Overloading - Example

```
class Account{
 int acc_no;
 String name;
 float amount;
 void insert(int a, String n, float amt){
   acc_no = a;
   name = n;
   amount = amt;  }
 void insert(int a, String n){
   acc_no = a;
   name = n;
   amount = 1000;  }
 void display(){
   System.out.println(acc_no+" "+name+" "+amount);}
}
```

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

      Account a2 =  new Account();
      a2.insert(832346,“Shobit“);
      a2.display();
  }
}
```

Output:
832345 Ankit 5000.0
832346 Shobit 1000.0

Can Main() be overloaded?


```
public static void main(String[] args){
   System.out.println("main with String[]");
}

public static void main(String args){
   System.out.println("main with String");
}

public static void main(){
   System.out.println("main without args");
}
```

Ans: Yes. But, JVM calls main() method which receives **string array** as arguments only.

What should be printed?


```
class OverloadingCalculation{
  void add(int a, int b){
       System.out.println("int arg method invoked");
  }
  void add(long a, long b){
       System.out.println("long arg method invoked");
  }

  public static void main(String args[]){
       OverloadingCalculation obj = new OverloadingCalculation();
       obj.add(20, 20);
  }
}
```

Output:
int arg method invoked

What should be printed?


```
class OverloadingCalculation{
  void add(int a, long b){
       System.out.println("a method invoked");
  }
  void add(long a, int b){
       System.out.println("b method invoked");
  }

  public static void main(String args[]){
       OverloadingCalculation obj = new OverloadingCalculation();
       obj.add(20, 20);
  }
}
```

Output:
Compile time error

Promotion Ambiguity

Constructor Overloading

- **Recall:** Constructor is just like a method but without return type.
- **Constructor overloading:** Having more than one constructor with different parameter lists.
- The compiler differentiates by the **number of parameters** in the list and their **types**.

Constructor Overloading - Example

```
class Account{
  int acc_no;
  String name;
  float amount;
  Account(int acc, String aname){
   acc_no = acc;
   name = aname;
   amount = 1000;
  }
  Account(int acc, String aname, float amt){
   acc_no = acc;
   name = aname;
   amount = amt;
  }
  void display(){
   System.out.println(acc_no+" "+name+" "+amount);
  }
}
```

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

Output:
832345 Ankit 5000.0
832346 Shobit 1000.0

Passing Objects to Constructors Example


```
class Account{
   int acc;
   String name;
   float amount;
   Account(int act, String aname){
     acc = act;
     name = aname;
   }
   Account(Account a){
     acc = a.acc;
     name = a.name;
   }
   boolean equalTo(Account a) {
     return(acc == a.acc && name == a.name);
   }
   void display(){
     System.out.println(acc+" "+name+" "+amount);}
}
```

```
class TestAccount{
  public static void main(String[] args){
    Account a1 = new Account(832345,"Ankit");
    Account a2 = new Account(a1);
    Account a3 = new Account(832346,"Shobit");
    System.out.println("a1==a2: " + a2.equalTo(a1));
    System.out.println("a1==a3: " + a3.equalTo(a1));
    a1.name = "Sahni";
    a1.display();
    a2.display();
  }
}
```

Output:

a1 == a2: true
a1 == a3: false
832345 Sahni 0.0
832345 Ankit 0.0

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You!