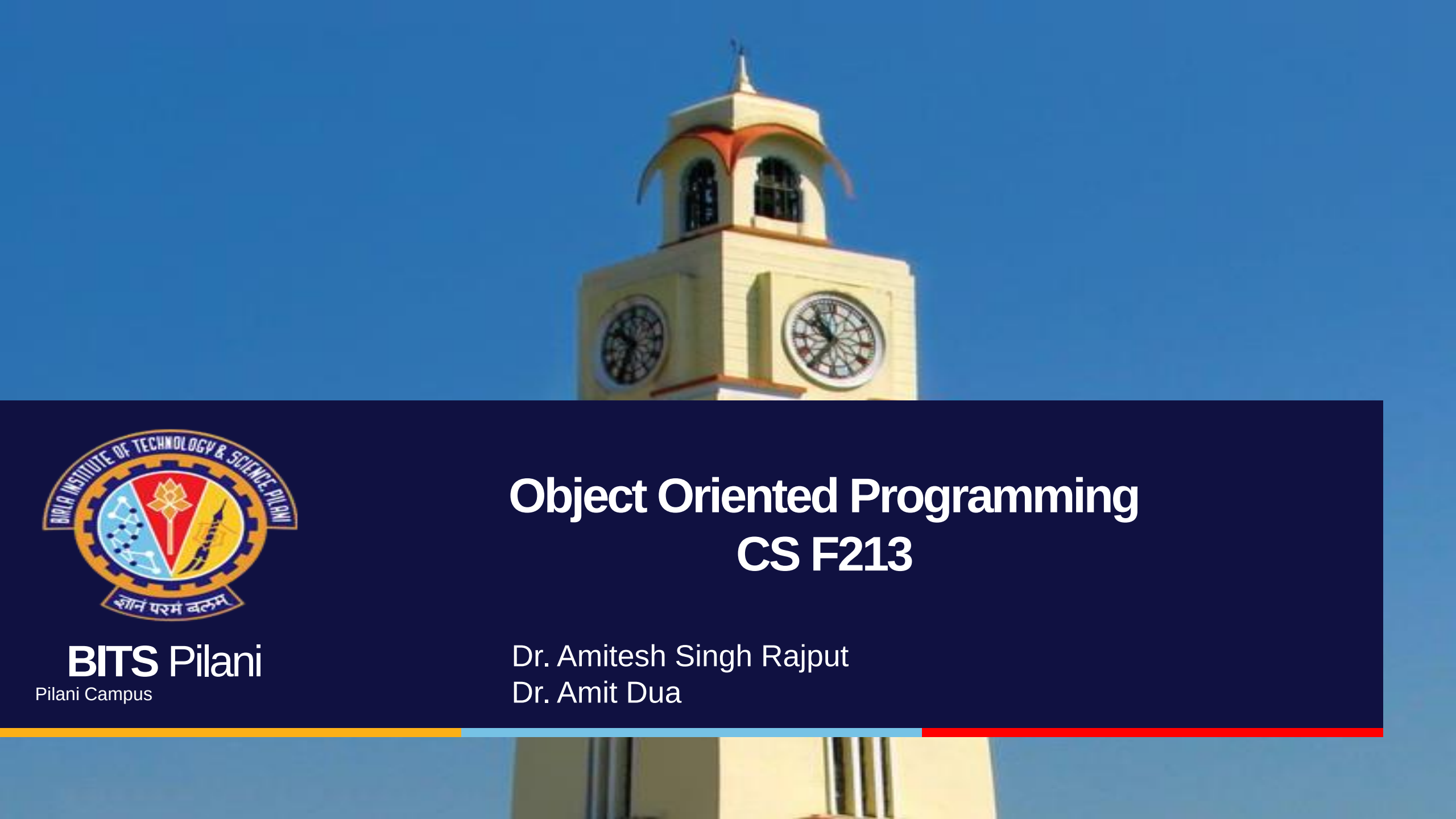


Object Oriented Programming
CS F213

BITS Pilani
Pilani Campus

Dr. Amitesh Singh Rajput
Dr. Amit Dua

Type Conversion and Type Casting Constructors

BITS Pilani

Pilani Campus

Primitive Data Types

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

Note: There is no `sizeof` operator in Java. The size of each primitive data type is fixed.

Type Conversion

- When the value of one data type is assigned to another, the two types might not be compatible with each other.
  - If the data types are compatible, then Java will perform conversion automatically known as Automatic Type Conversion.
  - If not then they need to be casted or converted explicitly.

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

Double –> Float –> Long –> Int –> Short –> Byte

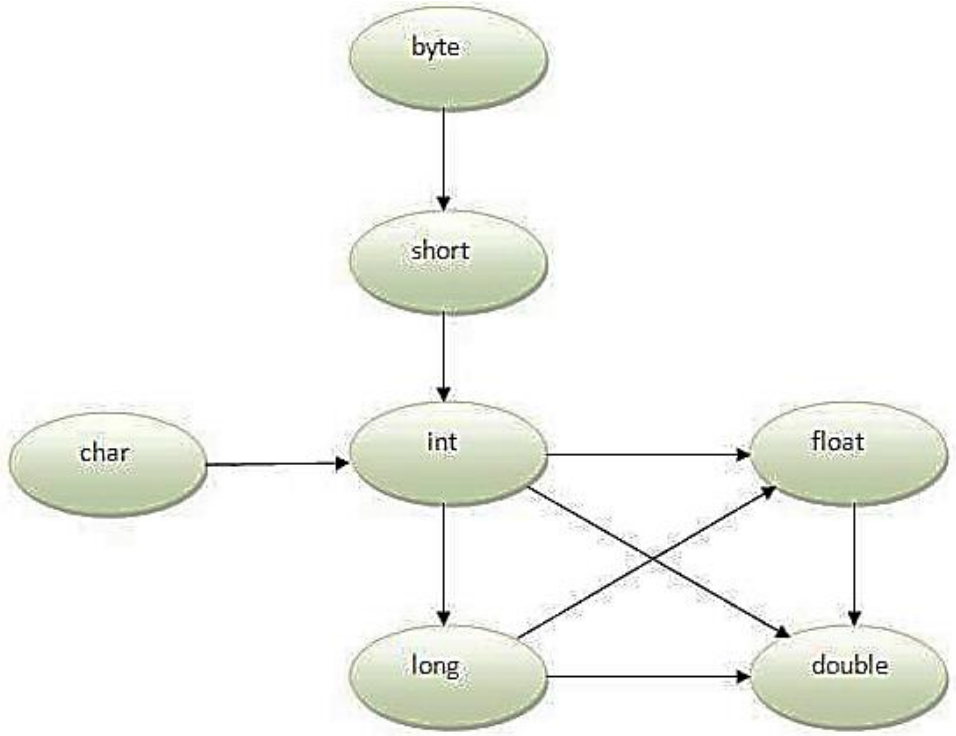
Narrowing or Explicit Conversion

Type Conversion - Example


```
char ch = 'c';
int num = 88;
num = ch;   // Automatic type conversion
System.out.println(num);
Ans: 99

ch = num;
System.out.println(ch);
  Error: incompatible types: possible lossy conversion from int to
  char

  Solution: ch = (char) num; //Explicit conversion

Ans: X
```

Type Promotion

- While evaluating expressions, the intermediate value may exceed the range of operands and hence the expression value will be promoted.
- For example,
  - If one operand is a long, float or double the whole expression is promoted to long, float or double respectively.



Type Promotion-Example

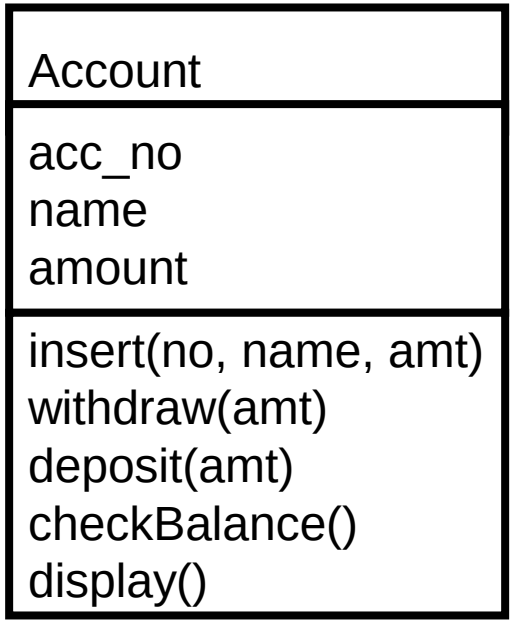

```
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = .1234;

(f * b) + (i / c) - (d * s);

double result = (f * b) + (i / c) - (d * s);
System.out.println("result = " + result);
```

Additional Example

```
byte b = 50;
b = (b * 2);
// byte gets promoted to store an integer value

System.out.println(b);
Error: due to type promotion

Solution: b = (byte) (b * 2);  //Explicit conversion
Ans:  100
```

Class & Object - Example

- Write a java program for the class diagram given below.




```
class Account{
  int acc_no;
  String name;
  float amount;

  void insert(int a,String n,float amt){
   acc_no=a;
   name=n;
   amount=amt;
  }
```

Constructors

- Similar to a method but it is called when an object is created. The memory is allocated for the object.
- Used to initialize an object.
- It is not necessary to write a constructor for a class, the compiler creates a default constructor.

More about constructors

Rules for creating constructor

- Name must be same as its class name.
- Must have no explicit return type.

Types of constructors

- Default (no argument) constructor
  - Provide default values to the object like 0, null etc.
- Parameterized constructor
  - Provide different values to distinct objects.

Default Constructor

Example 1


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

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

Output:
0 null 0.0

Default Constructor

Example 2

```
class Account{
  int acc_no;
  String name;
  float amount;
  Account(){
   System.out.println("The default values are:");
   amount = 1000;
  }
  void display(){
   System.out.println(acc_no +" "+ name +" "+ amount);
  }
}
class TestAccount{
  public static void main(String[] args){
   Account a1 = new Account();
   a1.display();
  }
}
```

Output:
The default values are:
0 null 1000.0

Parameterized Constructor

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;   }*/
  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();
   a1.insert(832345,"Ankit",5000);
   a1.display(); */
   Account a1 = new Account(832345,"Ankit",5000);
   a1.display();
  }
}
```

Output:
832345 Ankit 5000.0

Note: When parameterized constructors are implemented, the copy of the default constructor is not created. In this example, if you want to create an object as Account a1 = new Account();, the default constructor definition needs to be provided within the class.

Difference between a Constructor and method

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behavior of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name will not be same as the class name.

Recalling Inheritance

Syntax of Inheritance

To inherit a class in Java "**extends**" keyword is used.

Function Overloading: Defining the function again with

1. Same Name
2. **Different** Arguments(Number, Sequence, Type)
3. Same class or Base and Derived class

Function Overriding: Defining the function again with

1. Same Name
2. **Same** Arguments(Number, Sequence, Type)
3. Base and Derived Class

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>);
```