

# ARGUMENTS & NESTED CLASSES LECTURE 10B

---

Prof. Anita Agrawal,  
BITS, Pilani-K.K.Birla Goa Campus

# final keyword in java

- final is a non-access modifier applicable to:
  - Variable
  - Method
  - Class

**Final Variable** → To create constant variables

**Final Methods** → Prevent Method Overriding

**Final Classes** → Prevent Inheritance

- ***final*** variable:
  - When a variable is declared with ***final*** keyword, it's value cannot be modified.
  - It is essentially a constant.
- If reference:
  - cannot be re-bound to reference another object
  - Internal state of the object pointed can be changed

- The final variable has to be initialized, otherwise compile-time error
- Can be initialized only once

# Ways to initialize the final variable

- Initialization during declaration:

Example:

- final int num = 7;
- static final double length = 52.5
- If not initialised during declaration: Blank variable
- Blank variable initialisation:
  - inside the instance initialisation block
  - inside the constructor
  - inside all the constructors if more than one constructor
  - inside the static block if a static variable

//Java program to demonstrate different

// ways of initializing a final variable

```
class A
```

```
{
```

```
    // a final variable direct initialize
```

```
    final double width = 10.5;
```

```
    // a blank final variable
```

```
    final int sum;
```

```
    // another blank final variable
```

```
    final int height;
```

```
    // a final static variable PI direct initialize
```

```
    static final double PI = 3.14;
```

```
    // a blank final static variable
```

```
    static final double grav_const;
```

```
// instance initializer block for initializing sum
```

```
{  
    sum = 25;  
}
```

```
// static initializer block for initializing grav_const
```

```
static {  
    grav_const = 9.8;  
}
```

// Constructor for initializing height. Note that if there are more // than one constructors, you must initialize height in them also

```
public OOP()  
{  
    height = 20;  
}  
}
```

- As you know that a final variable cannot be re-assigned. But in case of a reference final variable, internal state of the object pointed by that reference variable can be changed.

Note: This is not re-assigning. This property of *final* is called *non-transitivity*.

- The *non-transitivity* property also applies to arrays., final arrays.



// Java program to demonstrate reference final  
variable

- A final variable cannot be reassigned, doing it, will throw compile-time error.

Java program to demonstrate re-assigning final variable will  
//throw compile-time error

.....finref2

When a final variable is created inside a method/constructor/block, it is called **local final variable**, and it must be initialized once where it is created.

## Java program to demonstrate local final variable

```
class A
{
    public static void main(String args[])
    {
        // local final variable
        final int x;
        x = 15;
        System.out.println(x);
    }
}
```

# Difference between a normal variable and a final variable

- Values can be reassigned to normal variables, but final variable values cannot be changed once assigned.
- Use final variables only for the values that are required to remain constant throughout the execution of program.

# Final classes

- Declared with *final* keyword.
- A final class cannot be extended(inherited).
- Uses of a final class :
  - To prevent inheritance, as final classes cannot be extended.
  - To create an immutable class like the predefined [String](#) class.
    - You can not make a class immutable without making it final.

# final method

- Declared with a final keyword.
- A final method cannot be overridden
- Declare methods with final keyword to follow the same implementation throughout all the derived classes.



//program to illustrate final keyword with the method

class A

```
{  
    final void m1()  
    {  
        System.out.println("This is a final method.");  
    }  
}
```

class B extends A

```
{  
    void m1()  
    {  
        // COMPILE-ERROR! Can't override.  
        System.out.println("Illegal!");  
    }  
}
```