Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 1

Lecture 7

Lecture 7

- **Static Member**
- **Final Member**

Static Keyword in Java

- The static keyword in Java is mainly used for memory management.
- The static keyword in Java is used to share the same variable or method of a given class.
- We can apply static keywords with variables, methods, blocks, and nested classes.
- The static keyword belongs to the class than an instance of the class.

The *static* keyword is a non-access modifier in Java that is applicable for the following:

1. Blocks
2. Variables
3. Methods
4. Classes

Note: *To create a static member(block, variable, method, nested class), precede its declaration with the keyword static.*

Characteristics of static keyword:

- **Shared memory allocation:** Static variables and methods are allocated memory space only once during the execution of the program. This memory space is shared among all instances of the class, which makes static members useful for maintaining global state or shared functionality.
- **Accessible without object instantiation:** Static members can be accessed without the need to create an instance of the class. This makes them useful for providing utility functions and constants that can be used across the entire program.

- **Associated with class, not objects:** Static members are associated with the class, not with individual objects. This means that changes to a static member are reflected in all instances of the class, and that you can access static members using the class name rather than an object reference.
- **Cannot access non-static members:** Static methods and variables cannot access non-static members of a class, as they are not associated with any particular instance of the class.

- **Can be overloaded, but not overridden**: Static methods can be overloaded, which means that you can define multiple methods with the same name but different parameters. However, they cannot be overridden, as they are associated with the class rather than with a particular instance of the class.

```
class Test
{
    // static method
    static void m1()
    {
        System.out.println("from m1");
    }
    public static void main(String[] args)
    {
         // calling m1 without creating
         // any object of class Test
          m1();
    }
}
```

Static blocks

- If you need to do the computation in order to initialize your static variables, you can declare a static block that gets executed exactly once, when the class is first loaded.

```
class Test
{
    // static variable
    static int a = 10;
    static int b;
    // static block
    static {
        System.out.println("Static block initialized.");
        b = a * 4;
    }
     public static void main(String[] args)
    {
       System.out.println("from main");
       System.out.println("Value of a : "+a);
       System.out.println("Value of b : "+b);
    }
}
```

Static variables

- When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level.
- Static variables are, essentially, global variables.
- All instances of the class share the same static variable.

Important points for static variables:

- We can create static variables at the class level only.
- static block and static variables are executed in the order they are present in a program.

Static methods

When a method is declared with the static keyword, it is known as the static method. The most common example of a static method is the main( ) method.

Methods declared as static have several restrictions:

- They can only directly call other static methods.
- They can only directly access static data.

Java static method

If you apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- static method can access static data member and can change the value of it.

Static Classes

- A class can be made static only if it is a nested class.
- We cannot declare a top-level class with a static modifier but can declare nested classes as static.
- Such types of classes are called Nested static classes.

```
class OuterClass
{
    private static String msg = "ABES Engineering College";
    public static class NestedStaticClass
     {
        public void printMessage()
        {
                System.out.println("Message " + msg);
        }
      }
}
```

```
class MyMain {
        public static void main(String args[])
        {
OuterClass.NestedStaticClass printer= new
OuterClass.NestedStaticClass();
                printer.printMessage();
        }
}
```

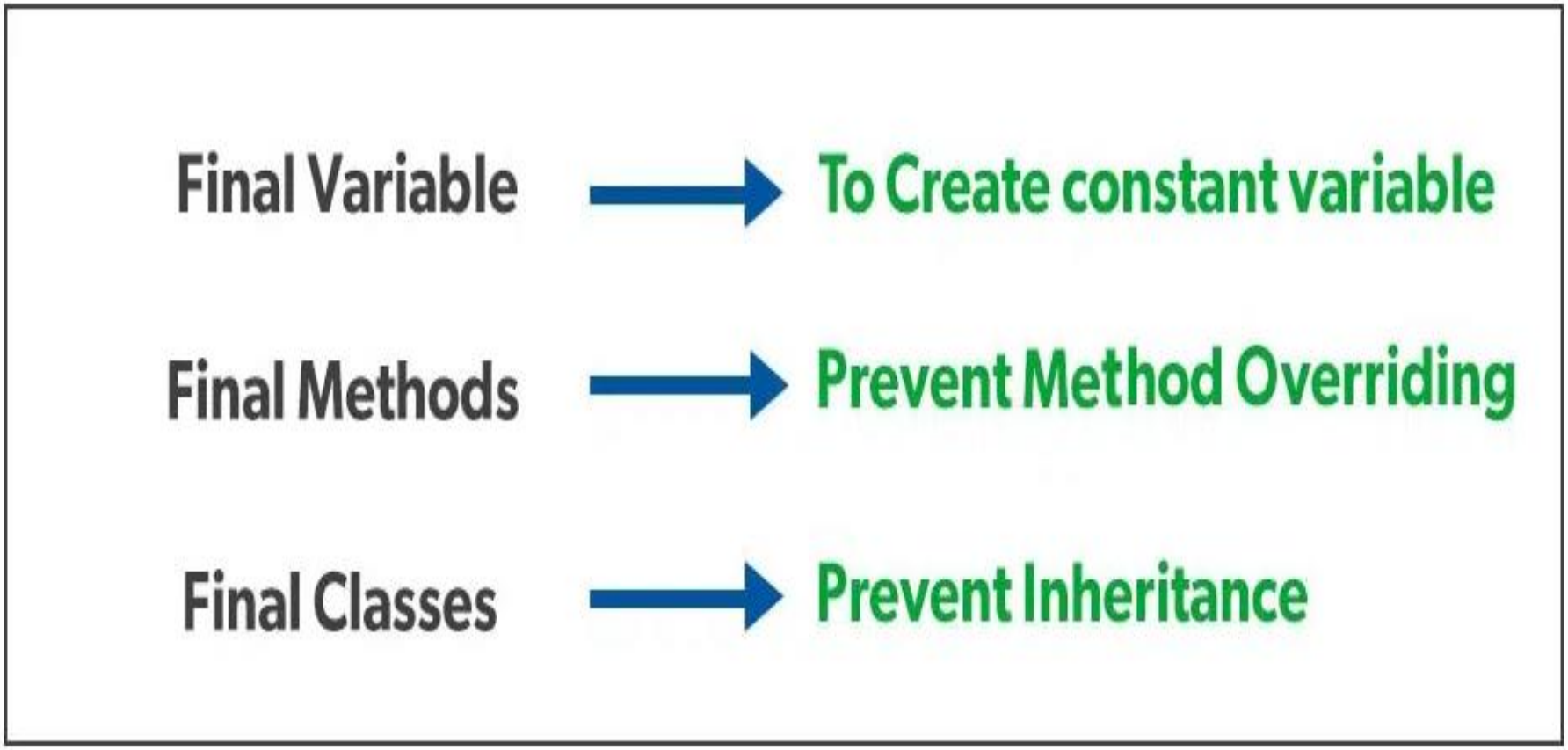
Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context.

Final can be:

- variable
- method
- class

- The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable.
- It can be initialized in the constructor only.
- The blank final variable can be static also which will be initialized in the static block only.


Final Variable
To Create constant variable
Final Methods
Prevent Method Overriding
Final Classes
Prevent Inheritance

If you make any variable as final, you cannot change the value of final variable(It will be constant).

```
class Bike9{
 final int speedlimit=90;//final variable
 void run(){
  speedlimit=400;
 }
 public static void main(String args[]){
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

Output:Compile Time Error

Java final method

If you make any method as final, you cannot override it.

```
class Bike{
  final void run(){System.out.println("running");}
}

class Honda extends Bike{
   void run(){System.out.println("running safely with 100kmph
    ");}

   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

Output:Compile Time Error

Java final class

If you make any class as final, you cannot extend it.

```
final class Bike{}

class Honda1 extends Bike{
  void run(){System.out.println("running safely with 100k
    mph");}

  public static void main(String args[]){
  Honda1 honda= new Honda();
  honda.run();
  }
}
```

Output:Compile Time Error

- A final variable that is not initialized at the time of declaration is known as blank final variable.
- If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful.

Can we initialize blank final variable?

Yes, but only in constructor. For example:

```
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
 }
}
```