Access Control and Static Keyword
Lecture 8

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

Controlling access to members of a class

- Access level modifiers determine whether other classes can use a particular field or invoke a particular method.
- Two levels of access control:
  - At the top level—public, or *package-private* (no explicit modifier).
  - At the member level—public, private, protected, or *package-private* (no explicit modifier).
- Class modifiers:
  - Public: The class is visible to all classes everywhere
  - Package-Private: (default): visible only within its own package

- Class Member modifiers:
  - Public modifier: The member is visible everywhere
  - Package Private: The member is visible only within the package in which it has been defined
  - Protected: The member can only be accessed within its own package (as with *package-private*) and, in addition, by a subclass of its class in another package
  - Private: the member can only be accessed in its own class.

Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

Access levels, why???

- Access levels affect you in two ways.
  - First, when you use classes that come from another source, such as the classes in the Java platform, access levels determine which members of those classes your own classes can use.
  - Second, when you write a class, you need to decide what access level every member variable and every method in your class should have.

Tips on choosing access level

- If other programmers use your class, you want to ensure that errors from misuse should not happen. Access levels can help you do this.
  - Use the most restrictive access level that makes sense for a particular member. Use private unless you have a good reason not to.
  - Avoid public fields except for constants. Public fields tend to link you to a particular implementation and limit your flexibility in changing your code.

Encapsulation

- Access Control is an important aspect of Encapsulation
- It is the mechanism that binds together code and the data it manipulates

Another Example

```
//Java program to illustrate using class from different class with
 private modifier
class Honda
{
private void display()
    {
        System.out.println("Private");
    }
}

class B
{
public static void main(String args[])
    {
        Honda  amaze = new Honda();
    //trying to access private method of another class
         amaze.display();
    }
}
```

Output-

error: display() has private access in Honda

amaze.display();

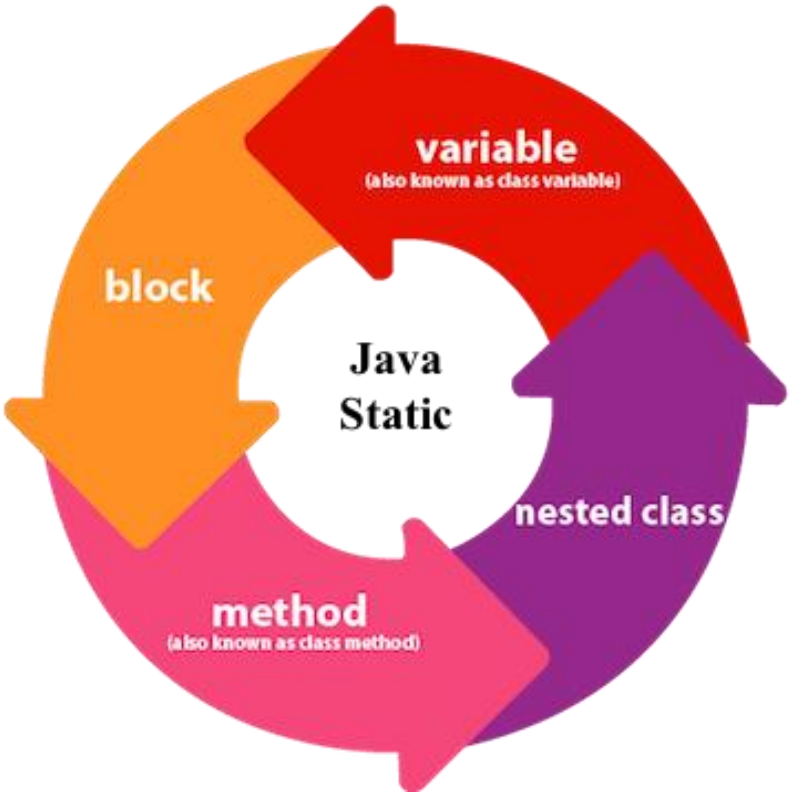
Static modifier

- When a number of objects are created from the same class, they each have their own distinct copies of *instance variables*.
- Each object will have its own values for these variables, stored in different memory locations.
- Sometimes, we want to have variables that are common to all objects.
  - Use *static modifier*
  - Static variables are associated with the class, rather than with any object.
  - Every instance of the class shares the class variable, which is in one fixed location in memory.

Static keyword

- The **static keyword** is used mainly for memory management.

- It is applicable to
  - blocks
  - variables
  - nested classes
  - methods



Static Keyword

- It precedes variable declaration with the keyword static and is common to all objects.

- Ways of intialisation:
  - Directly: modifier(if any) static var_type var_name;

    Example: public static int x =2;

or…….

  - Declare a static block

    Example: static int x;

```
static {
        x = 2;
       }
```

- Any object can change the value of the class variable, but class variables can also be manipulated without creating an instance of the class.
  - Syntax: class_name . var_name = value;
  - Example: class A. x = 3;

Static Methods

- Methods can also be declared as static.
  - For Example in public static void main() – main() can be called without creating an object

- Restrictions
  - They can only access static data
  - They can call only other static methods
  - They can't refer to this or super(Inheritance)