# Access Modifiers

Private, Default, Protected & Public

# Access Modifiers

- The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class.
- We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.
- There are four types of access modifiers available in java:
  - ✓Private
  - ✓Default (No keyword required)
  - ✓Protected
  - ✓Public
- Only one accessibility modifier can be specified for a member.

# Private

- The private access modifier is accessible only within the class.
- When a member of a class is specified as **private**, then that member can only be accessed by other members of its class.

```
class A {
        private int a;
        void show(int x) {
                a=x;
                System.out.println("a = " + a);
        }
}
class B
{
        public static void main(String args []) {
                A obj=new A();
                obj.show(10);
        }
}
```

# Private

- If you make any class constructor private, you cannot create the instance of that class from outside the class.

```
class A {
        private A() {
                System.out.println("Hello, I am a private constructor" );
        }
}
class B
{
        public static void main(String args []) {
                A obj=new A();

        }
}
```

# Private

- If you make any class constructor private, you cannot create the instance of that class from outside the class.

```
class A {
        private A() {
                System.out.println("Hello, I am a private constructor" );
        }
}
class B
{
        public static void main(String args []) {
                A obj=new A();

        }
}
```

**Output:**
Error-
A() has private access in A

# Public

- The **public** access modifier is accessible everywhere.
- It has the widest scope among all other modifiers.
- **Public** accessibility is the least restrictive of all the accessibility modifiers.

# Default

- When no access modifier is used, then by default the member of a class is public within its own package, but cannot be accessed outside of its package.
- It provides more accessibility than private. But, it is more restrictive than protected, and public.

# Protected

- A **protected** member is accessible in all classes in the same package, and by all subclasses of its class in any package where this class is visible.
- In other words, non-subclasses in other packages cannot access protected members from other packages.
- A subclass in another package can only access **protected** members in the superclass via references of its own type or its subtypes.
- It is more restrictive than **public** member accessibility.

# Protected

- Example:

```
package packA;
class A {
        private int a;
        protected void show() {
                System.out.println("Hello, I am protected method" );
        }
}

package packB;
import packA.A;
class B extends A
{
        public static void main(String args []) {
                A obj=new A();
                obj.show();
        }
}
```

# Protected

- Example:

```
package packA;
class A {
        private int a;
        protected void show() {
                System.out.println("Hello, I am protected method" );
        }
}

package packB;
import packA.A;
class B extends A
{
        public static void main(String args []) {
                B obj=new B();
                obj.show();
        }
}
```

# Access Modifiers- Accessibility

| Access Modifier | within class | within package | outside package by subclass only | outside package |
|---|---|---|---|---|
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

# Access Modifiers with Method Overriding

- If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.

```
class A {
        protected void show() {
                System.out.println("Hello, I am protected method in class A" );
        }
}

class B extends A
{
        void show() {
                System.out.println("Hello, I am default method in class B" );
        }

        public static void main(String args []) {
                A obj=new A();
                obj.show();
        }
}
```