JAVA ACCESS MODIFIERS

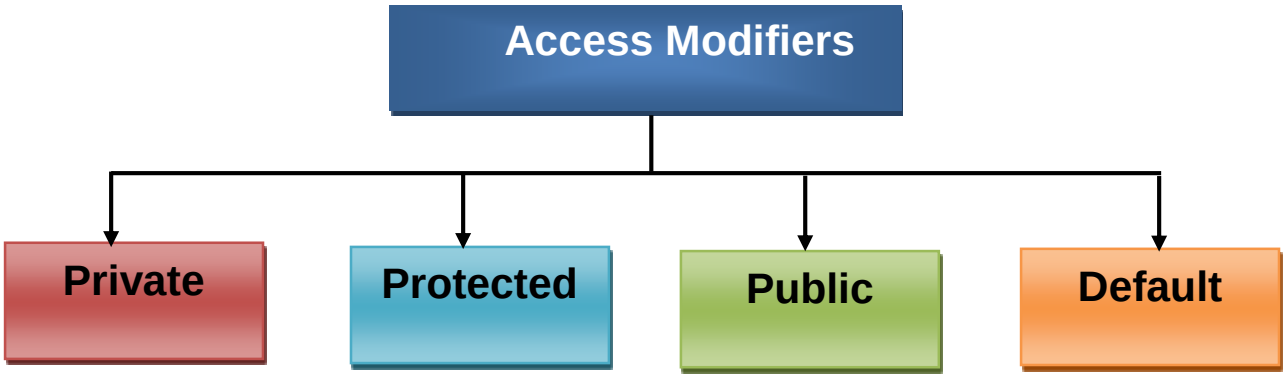
ACCESS MODIFIERS

- It is mainly used to decide the scope of various programming elements such as variables, functions, constructor, destructor, etc, ..
- It gives the scope and life time to object properties (instance variables & functions) and class properties (static variables & functions)
- It is used to access the state (e.g. variable) and behavior (e.g function) of objects / class
- It is used to set the access privilege levels to members of a class (e.g. variables/functions)
- Java supports four different types of modifiers. They are:
    1. Private
    2. Public
    3. Protected
    4. Default / No Modifier / Package Private (Default for variables / methods / constructors / classes)

Usage

- It is used to implement the important feature of oops called as **data hiding**

ACCESS MODIFIERS



ACCESS LEVELS

S. N	Access Modifiers	Same Package			Other Package		
		Same class in same package	Sub class in same package	Other class in same package	Same class in other package	Sub class in other package	Other class in other package
1	Private	Yes	No	No	No	No	No
2	Public	Yes	Yes	Yes	Yes	Yes	Yes
3	Protected	Yes	Yes	Yes	No	Yes	No
4.	Default (Package Private)	Yes	Yes	Yes	No		

GENERAL SYNTAX OF ACCESS MODIFIERS

```
class <name>
{
        private members;                    (variables & methods)
        public members:                     (variables & methods)
        protected members;                  (variables & methods)
        default members;                    (variables & methods)
}
```

Example

```
class Test
{
        private int number;
        private char address[100];
        protected char name[100];
        public void disp()
        {
                // user code
        }

}
```

1. PRIVATE MODIFIER

- It is accessible only within a same class (current class) of same package
- Private data can't be accessible in outside of a class or other packages
- It gives more security than other modifiers

Usage

- It is used to implement the concept of data hiding (information hiding) feature

Syntax

```
<modifier> <type> <name>=value;
```

Example

```
private int a=95;
```

1. EXAMPLE OF PRIVATE MODIFIER

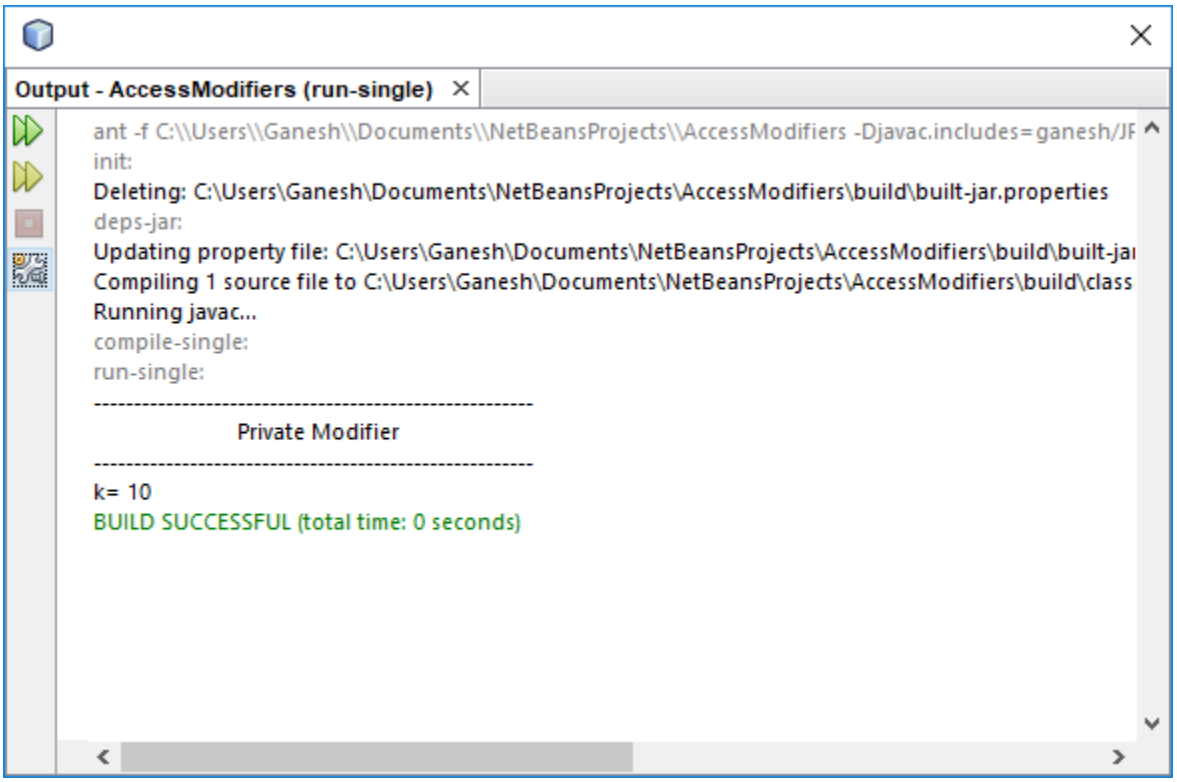
(JPrivateM.java)

SOURCE CODE

```
public class JPrivateM
{
// private variable definition
   private int k=10;
   public void disp()
   {
         System.out.println("k= "+k);
```

```
    }
    public static void main(String[] args)
    {
        System.out.println("-----------------------------------------------------");
        System.out.println("\tPrivate Modifier");
        System.out.println("-----------------------------------------------------");
// object creation
        JPrivateM obj=new JPrivateM();
// calling instance method using object
        obj.disp();
    }
}
```

2. OUTPUT



DRAWBACKS

- It is not possible to access the private variables in outside of a class.

2. PUBLIC MODIFIER

- It is accessible anywhere in the programming
- Most accessible modifier (recommended modifier)
- Public data can be accessible in outside of a class / main function / derived class (if inheritance involved)
- It does not provide security than other modifiers

Syntax

```
<modifier> <type> <name>=value;
```

Example

```
public String sname=”Vijay”;
```

2. EXAMPLE OF PUBLIC MODIFIER

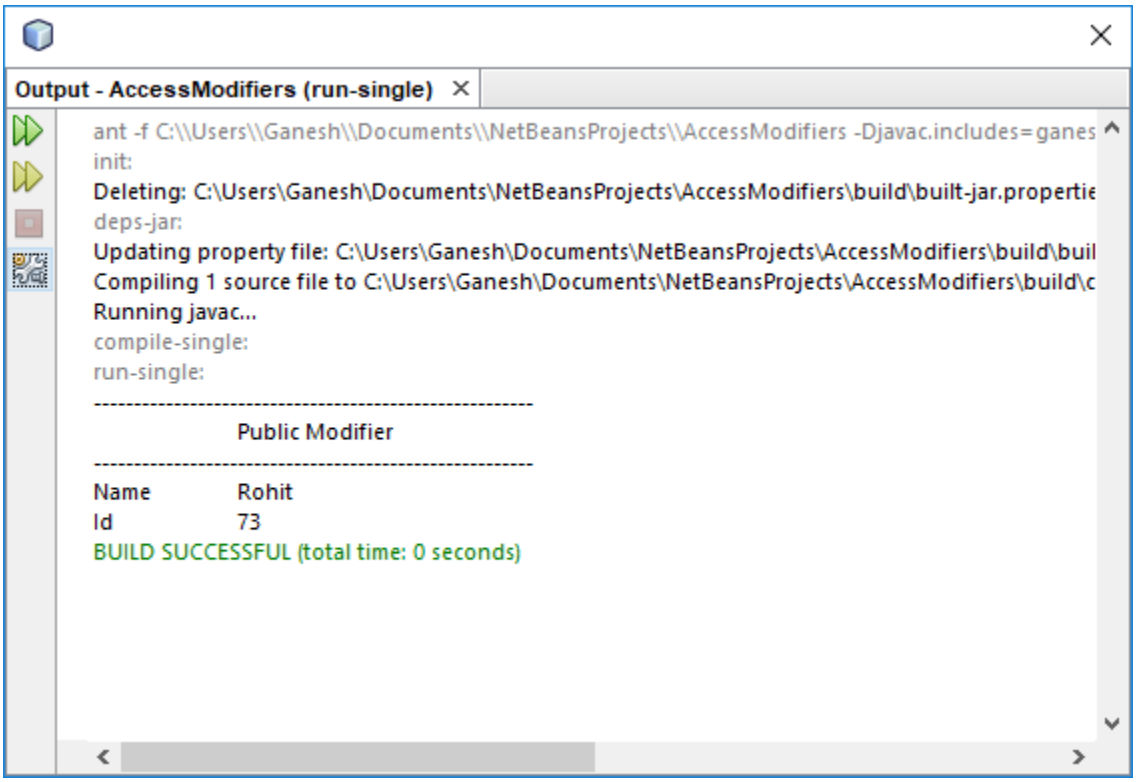
(JPublicM.java)

1. SOURCE CODE

```
public class JPublicM
{
// public variables definition
   public String name="Rohit";
   public int id=73;
   void disp()
   {
         System.out.println("Name \t"+name);
         System.out.println("Id \t"+id);
```

```
    }
    public static void main(String[] args)
    {
        System.out.println("-----------------------------------------------------");
        System.out.println("\tPublic Modifier");
        System.out.println("-----------------------------------------------------");
// object creation
        JPublicM obj=new JPublicM();
// calling instance method using object
        obj.disp();
    }
}
```

2. OUTPUT



3. PROTECTED MODIFIER

- It is similar to private modifier
- It is accessible anywhere within a same package (current class or sub class or other classes in same package) as well as in derived class in other package (If inheritance is involved)
- Protected data can't accessible in other class in other package

Syntax

```
<modifier> <type> <name>=value;
```

Example

```
protected int a=95;
```

3. EXAMPLE OF PROTECTED MODIFIER

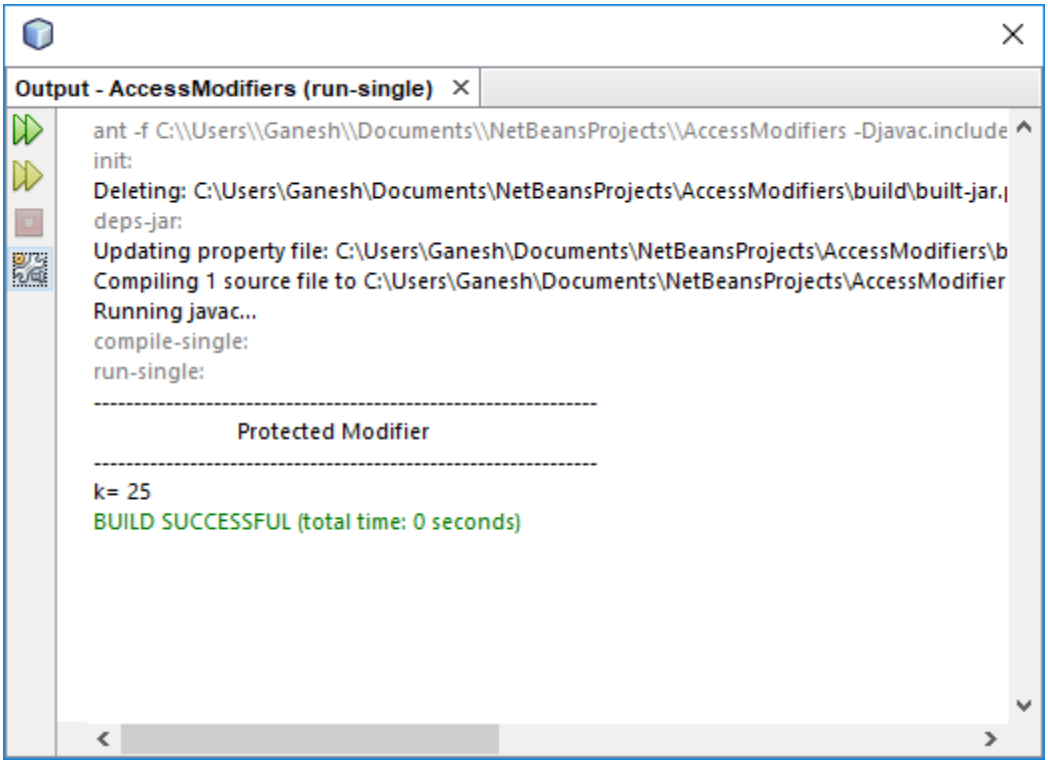
(JProtectedM.java)

1. SOURCE CODE

```
public class JProtectedM
{
// protected variable definition
   protected int k=25;
   public void disp()
   {
      System.out.println("k= "+k);
   }
   public static void main(String[] args)
```

```
    {
        System.out.println("-----------------------------------------------------------");
        System.out.println("\tProtected Modifier");
        System.out.println("-----------------------------------------------------------");
// object creation for current class
        JProtectedM obj=new JProtectedM();
// calling instance method using object
        obj.disp();
    }
}
```

2. OUTPUT



DRAWBACKS

- It is not possible to access the protected variables in other class of other package except sub class (derived class) in other package.

4. DEFAULT (PACKAGE PRIVATE | FRIENDLY)

- It is the default modifier in java and its scope is limited to current package only
- If we don't mention any modifier, then system will treat it as a default modifier (no modifier / package private)
- It can be accessible anywhere in the same / current package. But outside of the package is not possible.

Syntax

```
<type> <name>=value;
```

Example

```
int a=95;
```

4. EXAMPLE OF PACKAGE PRIVATE MODIFIER

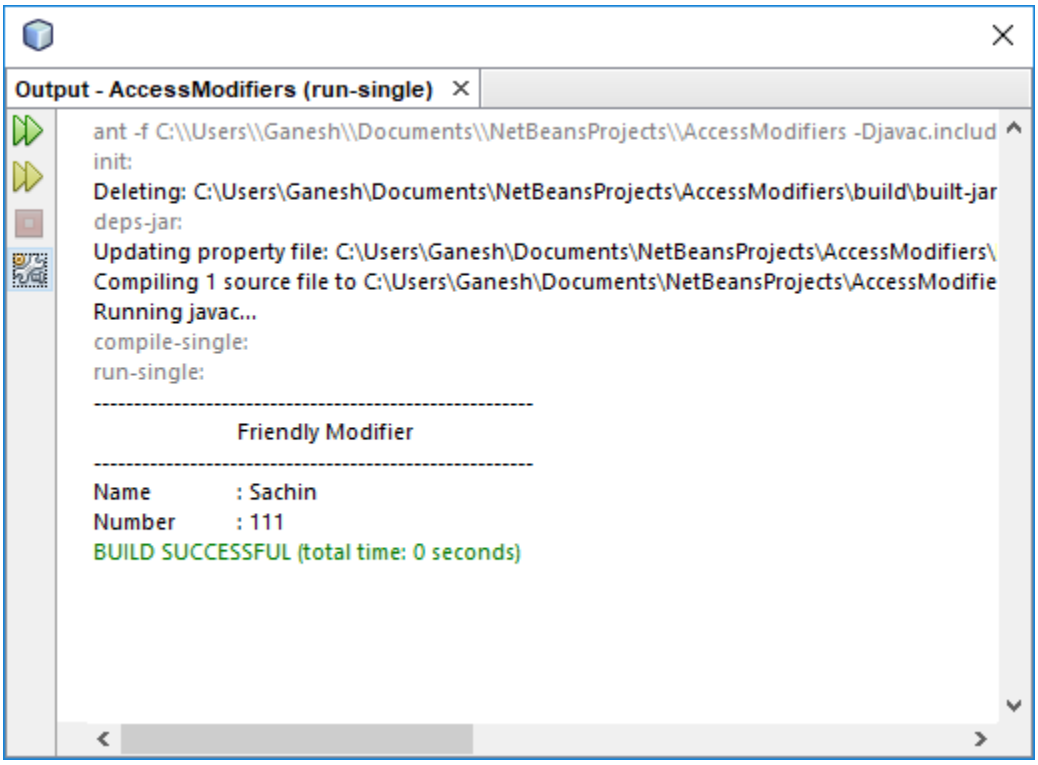
(JProtectedM.java)

1. SOURCE CODE

```
public class JPackagePrivateM
{
// package private (friendly) variable definition
   int n=111;
   String name="Sachin";
// package private (friendly) method implementation
   void info()
   {
         System.out.println("Name \t: "+name);
         System.out.println("Number \t: "+n);
```

```
    }
    public static void main(String[] args)
    {
        System.out.println("-----------------------------------------------------");
        System.out.println("\tFriendly Modifier");
        System.out.println("-----------------------------------------------------");
// object creation
        JPackagePrivateM obj=new JPackagePrivateM();
// calling instance method using object
        obj.info();
    }
}
```

2. OUTPUT



NOTE

- All the variables and method of a class is package private by default. (If no access modifier is specified)

SUMMARY

S.N	MODIFIERS	PURPOSE
1.	Private	It is accessible only within a class of same package
2.	Protected	It is accessible only within a class as well as derived class
3.	Public	It is accessible anywhere in the programming (including same / other packages)
4.	Package Private	It is accessible anywhere in the same package Default modifier

Default Access Modifiers

Variables - default modifier by default

Methods, Constructors - default modifier by default

Class - default modifier by default

Class Modifier

- In java, the default modifier of class is default modifier
- Class does not support private and protected modifiers. It supports only public and default modifiers.