In **Java**, an **access modifier** is a keyword used to set the visibility or accessibility of classes, methods, constructors, and variables. It controls which parts of a program can interact with the specified component. Java has four types of access modifiers

Java public keyword

A Java public keyword is an access modifier. It can be assigned to variables, methods, constructors, and classes. It is the most non-restricted type of access modifier.

Points to remember

- The public access modifier is accessible everywhere. So, we can easily access the public inside and outside the class and package.
- If you are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive. So, if you assign public to any method or variable, that method or variable can be overridden to sub-class using public access modifier only.
- If a program contains multiple classes, at most one class can be assigned as public.
- If a class contain a public class, the name of the program must be similar to the public class name.

Examples of public keyword

Example 1

Let's see an example to determine whether public variable and method is accessible or not outside the class. Here, we also try to create an instance of constructor outside the class.

```
class A {

  public String msg="Try to access a public variable outside the class";
   String info;
   public void display()
   {
      System.out.println("Try to access a public method outside the
class");
```

```
        System.out.println(info);
    }

    public A(String info)
    {
        this.info=info;
    }

}

public class PublicExample1 {
    public static void main(String[] args) {
        A a=new A("Try to create the instance of public constructor outside
the class");
        System.out.println(a.msg);
        a.display();

    }
}
```

Output:

Try to access a public variable outside the class

Try to access a public method outside the class

Try to create the instance of public constructor outside the class

Example 2

Let's see an example to determine whether public variable and method is accessible or not outside the package. Here, we also try to create an instance of constructor outside the package.

```
//save by A.java
package com.java;

public class A {
```

```
 public String msg="Try to access a public variable outside the
package";
 String info;
 public void display()
 {
    System.out.println("Try to access a public method outside the
package");
    System.out.println(info);
 }

 public A(String info)
 {
    this.info=info;
 }

}
//save by PublicExample1.java
package com.javatpoint;
import com.java.A;

public class PublicExample2 {
public static void main(String[] args) {
   A a=new A("Try to create the instance of public constructor outside
the package");
  System.out.println(a.msg);
  a.display();

}
}
```

Output:

Try to access a public variable outside the package

Try to access a public method outside the package

Try to create the instance of public constructor outside the package

Example 3

Let's see an example to determine whether the public method is overridden to sub-class using public access modifier.

```
class A
{
   public void msg()
   {
      System.out.println("Try it");
   }
}
class PublicExample3 extends A {
   public void msg()
{
   System.out.println("Try to access the overridden method");
}
public static void main(String[] args) {
   PublicExample3 p=new PublicExample3();
   p.msg();

}
}
```

Output:

Try to access the overridden method

Example 4

Let's see an example to determine whether the public method is overridden to sub-class using private access modifier.

```
class A
{
   public void msg()
   {
      System.out.println("Try it");
   }
}
```

```
class PublicExample4 extends A {
   private void msg()
{
   System.out.println("Try to access the overridden method");
}
public static void main(String[] args) {
   PublicExample4 p=new PublicExample4();
   p.msg();

}
}
```

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Cannot reduce the visibility of the inherited method from A

Example 5

Let's see an example to determine whether the public method is overridden to sub-class using default access modifier.

```
class A
{
   public void msg()
   {
      System.out.println("Try it");
   }
}

class PublicExample5 extends A {
   void msg()
{
   System.out.println("Try to access the overridden method");
}
public static void main(String[] args) {
   PublicExample5 p=new PublicExample5();
   p.msg();
```

```
}
}
```

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Cannot reduce the visibility of the inherited method from A

Example 6

Let's see an example to determine whether the public method is overridden to sub-class using protected access modifier.

```
class A
{
   public void msg()
   {
      System.out.println("Try it");
   }
}

class PublicExample6 extends A {
   protected void msg()
{
   System.out.println("Try to access the overridden method");
}
public static void main(String[] args) {
   PublicExample6 p=new PublicExample6();
   p.msg();

}
}
```

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Cannot reduce the visibility of the inherited method from A