MEHRAN UNIVERSITY
OF ENGINEERING & TECHNOLOGY
JAMSHORO, PAKISTAN

Object Oriented Programming

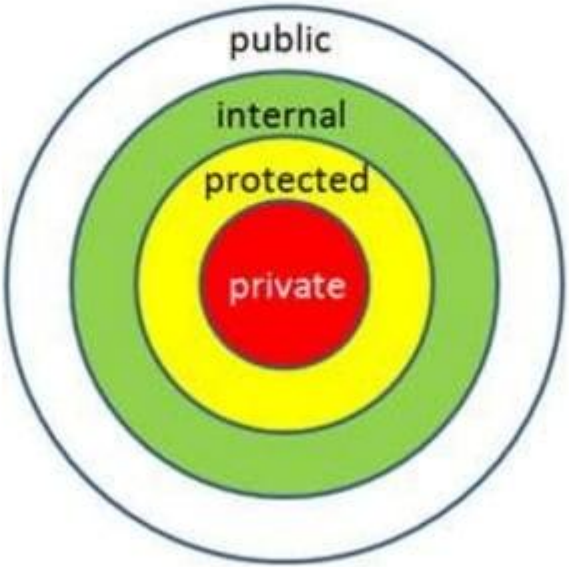
Introduction to Java

Lecture#02

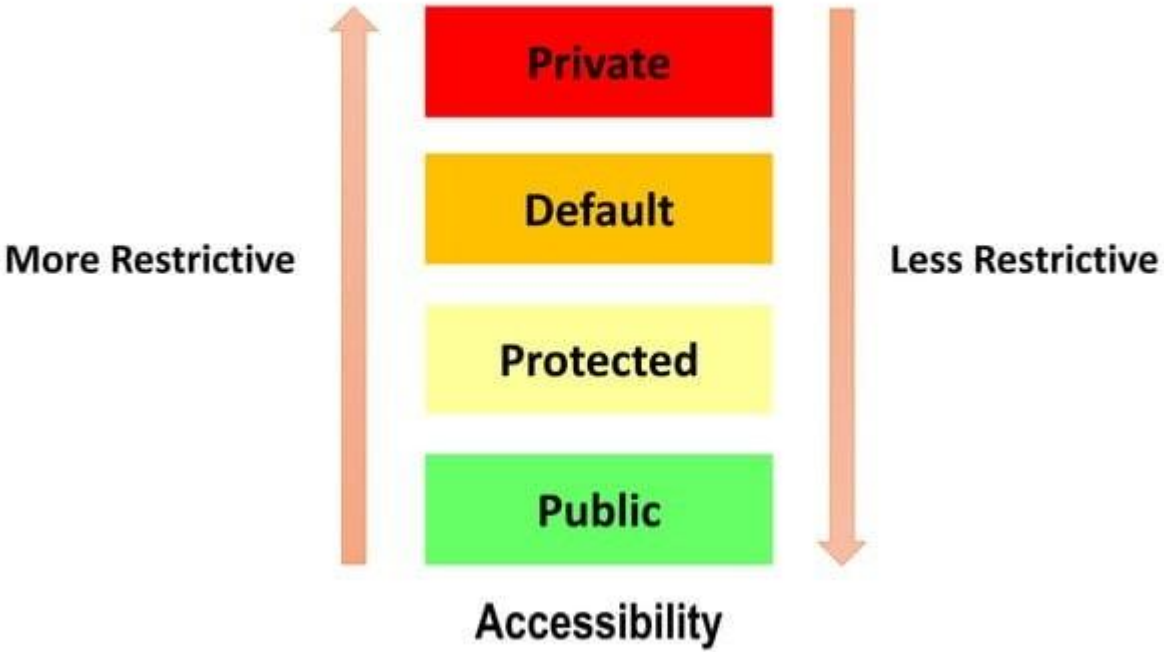
Access Modifiers

Lecture Slides by Engr. Mehwish Shaikh

Access Modifiers in java



Access Modifiers in java



Access Modifiers in java

- **Access Control Modifiers**
- Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are –
  - Visible to the package, the default. No modifiers are needed.
  - Visible to the class only (private).
  - Visible to the world (public).
  - Visible to the package and all subclasses (protected).

Access Modifiers in java

- **1) private Access Modifier**
  - The private access modifier is accessible only within class.
  - Methods, variables, and constructors that are declared private can only be accessed within the declared class itself.
  - Private access modifier is the most restrictive access level. Class and interfaces cannot be private.
  - Variables that are declared private can be accessed outside the class, if public getter methods are present in the class.
  - Using the private modifier is the main way that an object encapsulates itself and hides data from the outside world.

Private Access Modifier

```
public class Test{
   private int data=40;
 public static void main(String args[]){
  Test ex = new Test();
  System.out.println("Data is: "+ex.data);


  }
}
```

Private Access Modifier

```
public class Test{
  private int data=40;
 public static void main(String args[]){
  Test ex = new Test();
  System.out.println("Data is: "+ex.data);


  }
}
```

Here the *data* variable of the Example class is private and this variable accessed from same class itself

Private Access Modifier

```
public class Test{
   private int data=40;
 public static void main(String args[]){
  Test ex = new Test();
  System.out.println("Data is: "+ex.data);


  }
}
```

Data variable is assess in the same class where it is defined

Private Access Modifier

```
public class Test{
   private int data=40;
 public static void main(String args[]){
  Test ex = new Test();
  System.out.println("Data is: "+ex.data);


  }
}
```

Output is:
Data is: 40

Private Access Modifier

```
public class Test{
   private int data;
 public static void main(String args[]){
   ex.data=40;
   Test ex = new Test();
   System.out.println("Data is: "+ex.data);

   }
}
```

Same Program as previous but single modification

Now we set the value of the variable here using the object of Example class

Output is:
Data is: 40

Private Access Modifier

```
class A{
   private int data=40;
   private void msg() {
   System.out.println("Hello java");}
   }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

Private Access Modifier

```
class A{
   private int data=40;
   private void msg() {
   System.out.println("Hello java");}
   }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

This is first class where we declare a private variable and define private method

Private Access Modifier

```
class A{
   private int data=40;
   private void msg() {
   System.out.println("Hello java");}
   }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

This is second class where we try to access private variable and method

Private Access Modifier

```
class A{
  private int data=40;
  private void msg() {
  System.out.println("Hello java");}
  }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

Instance variable is private here

Private Access Modifier

```
class A{
   private int data=40;
   private void msg() {
   System.out.println("Hello java");}
   }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

Private method is here

Private Access Modifier

```
class A{
   private int data=40;
   private void msg() {
   System.out.println("Hello java");}
   }
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);  //Compile Time Error
  obj.msg();  //Compile Time Error
  }
}
```

Here, the data variable of the A class is private, so there's no way for other classes to retrieve or set its value directly.

Access Modifiers in java

- **2) public Access Modifier**
  - The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.
  - A class, method, constructor, interface, etc. declared public can be accessed from any other class. Therefore, fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.
  - However, if the public class we are trying to access is in a different package, then the public class still needs to be imported. Because of class inheritance, all public methods and variables of a class are inherited by its subclasses.

public Access Modifier

```
class A{
  public int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

public Access Modifier

```
class A{
  public int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

public instance variable and accessible in other classes

public Access Modifier

```
class A{
  public int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

Output is:
Data is: 40

Access Modifiers in java

- **3) protected access modifier**
  - The **protected access modifier** is accessible within package and outside the package but through inheritance only.
  - The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
  - Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

protected Access Modifier

```
class A{
  protected int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

protected Access Modifier

```
class A{
  protected int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

Protected instance variable

protected Access Modifier

```
class A{
  protected int data=40;
}
public class Test{
 public static void main(String args[]){
  A obj=new A();
  System.out.println("Data is: "+obj.data);
  }
}
```

Access protected instance variable in other class

Output is:
Data is: 40

Access Modifiers in java

- **4) default access modifier**
  - Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc.
  - A variable or method declared without any access control modifier is available to any other class in the same package. The fields in an interface are implicitly public static final and the methods in an interface are by default public.
  - The default modifier is accessible only within package.