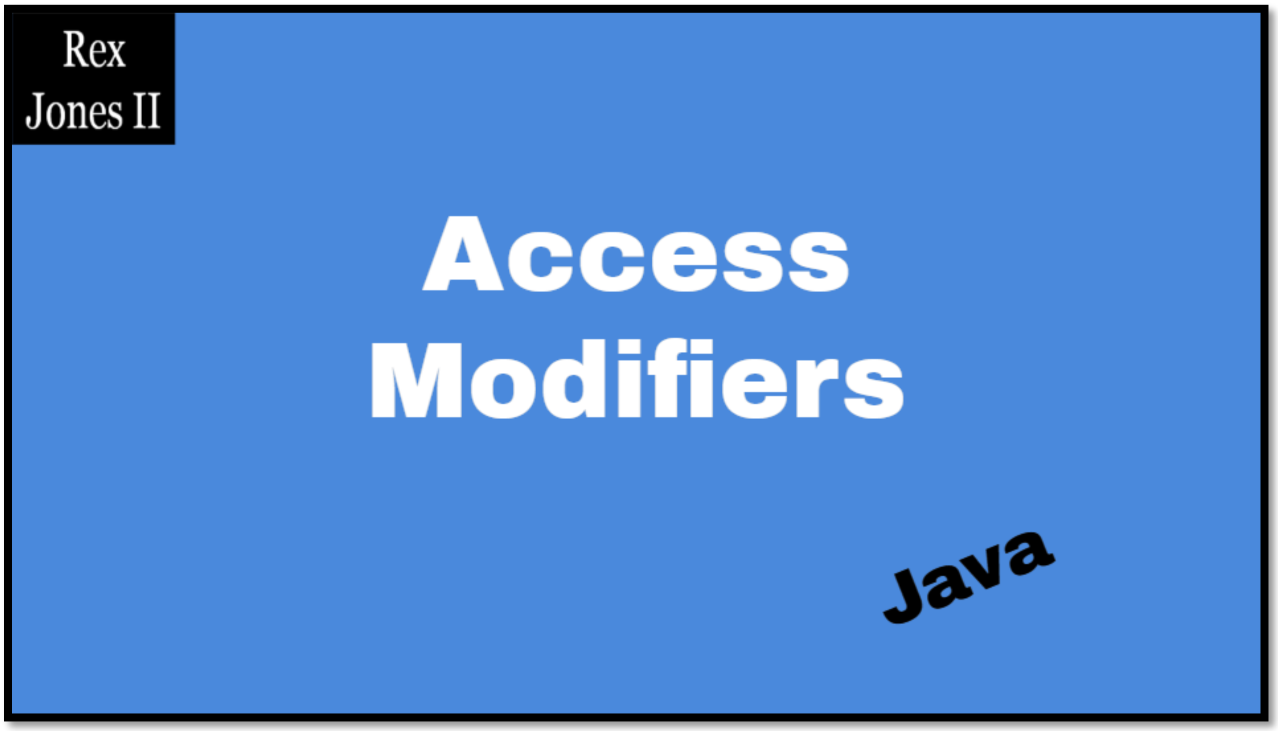
Rex Jones II

(Transcript) Access Modifiers

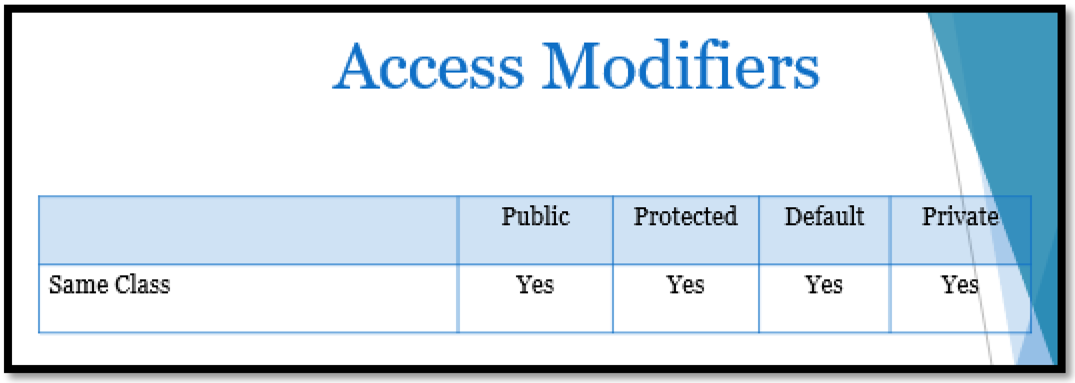


Introduction

The 4 Access Modifiers are public, protected, default, and private. This is the order of their access level. Public can be accessed by code outside its class. Protected is accessed by code outside its package and by code inside its package. If the code is outside its package then it must be inside of a sub class. Default can only be accessed by code in its package. Last is private which can only be accessed by code in its own class.

Let's pick back up where I left off with packages. A demopackage was created under java. Now, I'm going to create 1 more package called demopackage2. Right click java > New > Package > demopackage2. We are going to create 3 more classes. Right click demopackage1 > New > Class > Name will be Class2. Click the checkbox for public static void main. Right click demopackage2 > New > Class > Name will be Class3, click the checkbox public static void main and the 4th class will be under demopackage2. Name will be Class4 and click the checkbox again.

Same Class

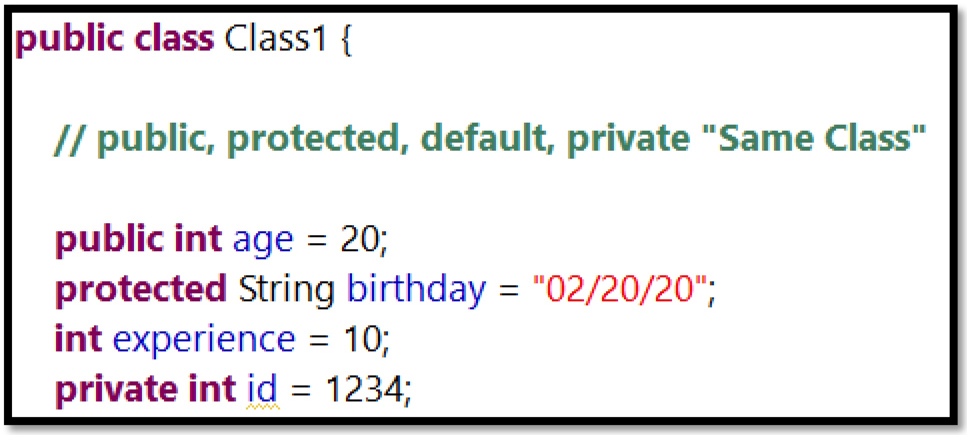


Access Modifiers

	Public	Protected	Default	Private
Same Class	Yes	Yes	Yes	Yes

In the same class, all access modifiers public, protected, default, and private are visible. For Class1, let's write our data variables according to their access level. As a comment, I will write // public, protected, default, private "Same Class".

First, is public int age = 20. Second, is protected String birthday = "02/02/2020". Third, is default which has no access modifier, int experience = 10, Fourth is private int id = 1234.

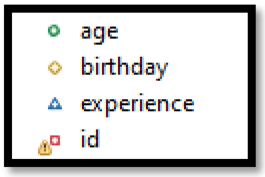


```
public class Class1 {

   // public, protected, default, private "Same Class"

   public int age = 20;
   protected String birthday = "02/20/20";
   int experience = 10;
   private int id = 1234;
```

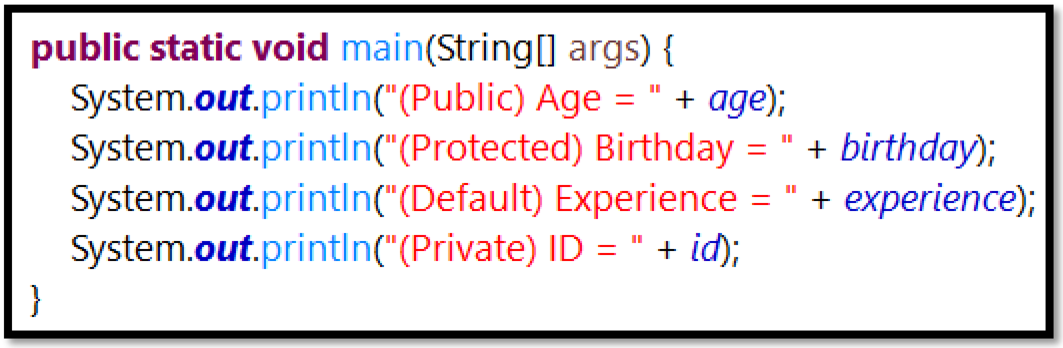
Did you notice the difference in symbols for each access modifier under Class1? Public is a green circle with a white dot. Protected looks like a yellow diamond. Default is a blue triangle and Private is a red square.



```
○ age
◇ birthday
△ experience
□ id
```

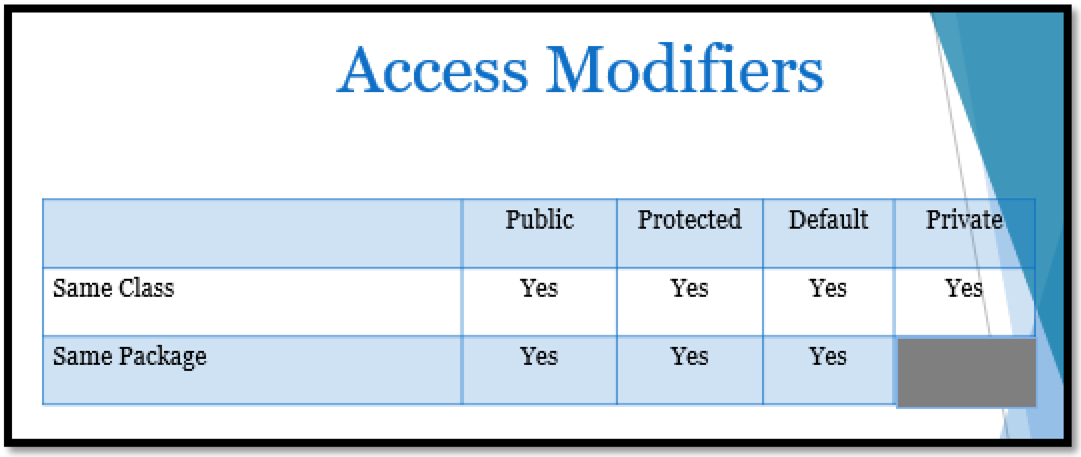
Let's add some print statements to show the access for each variable inside of a main method. Know what I have to make the variables static to print their value in the main method: static for age public, static for protected String, and default experience, and private id.

Starting with age, we are going to write. sysout("(Public) Age = " + age) / Next is sysout("(Protected) Birthday = " + birthday) / Third is default, sysout("(Default) Experience = " + experience) / Fourth is private, so we write sysout("(Private) ID = " + id). Let's Run and we see all values. Age, Birthday, Experience, and ID



```
public static void main(String[] args) {
   System.out.println("(Public) Age = " + age);
   System.out.println("(Protected) Birthday = " + birthday);
   System.out.println("(Default) Experience = " + experience);
   System.out.println("(Private) ID = " + id);
}
```

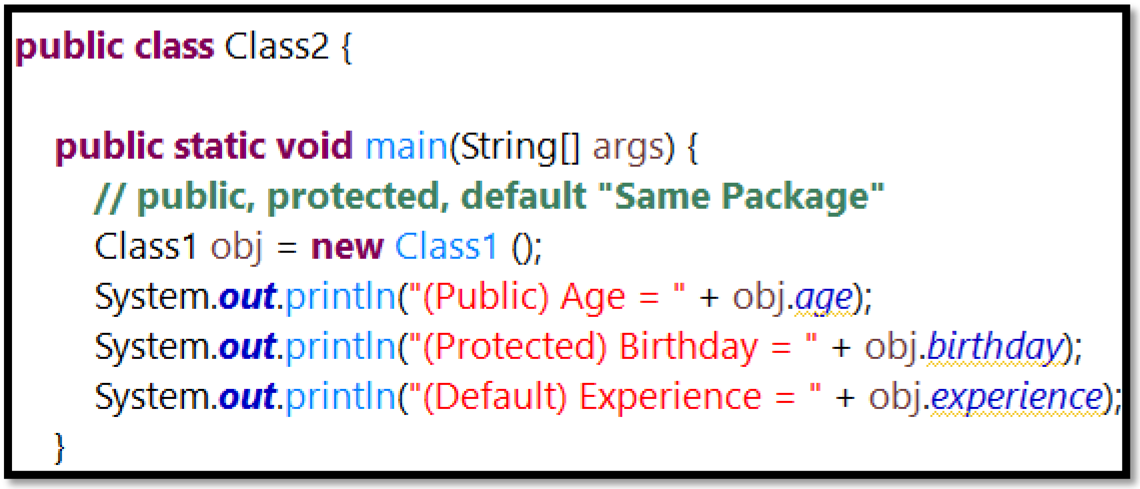
Same Package



Access Modifiers

	Public	Protected	Default	Private
Same Class	Yes	Yes	Yes	Yes
Same Package	Yes	Yes	Yes	▨

In the same package but different class, private will not visible. Let's copy the comment from Class1 and go to Class2 and paste it. Remove private and change Same Class to Same Package. There's no need to create an object because the variables are static. Know what let's go ahead and remove static from these variables. Also copy the print statements and make them a comment. Go to Class2 and create an object Class1 obj = new Class1 (). / obj. and we only see age, birthday, and experience. Private id is not available.

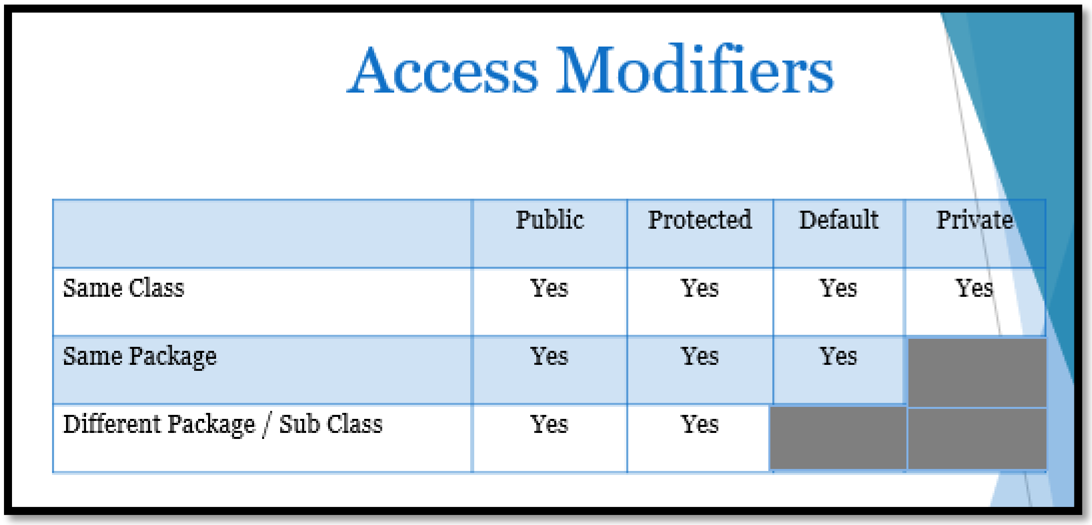


```
public class Class2 {

  public static void main(String[] args) {
    // public, protected, default "Same Package"
    Class1 obj = new Class1 ();
    System.out.println("(Public) Age = " + obj.age);
    System.out.println("(Protected) Birthday = " + obj.birthday);
    System.out.println("(Default) Experience = " + obj.experience);
  }
```

So we go ahead and add the print statements. Add obj in front of each variable. obj.age, obj.birthday, and obj.experience. There is no need to put obj in front of id because it's not visible. Let's Run. We see Age, Birthday, and Experience.

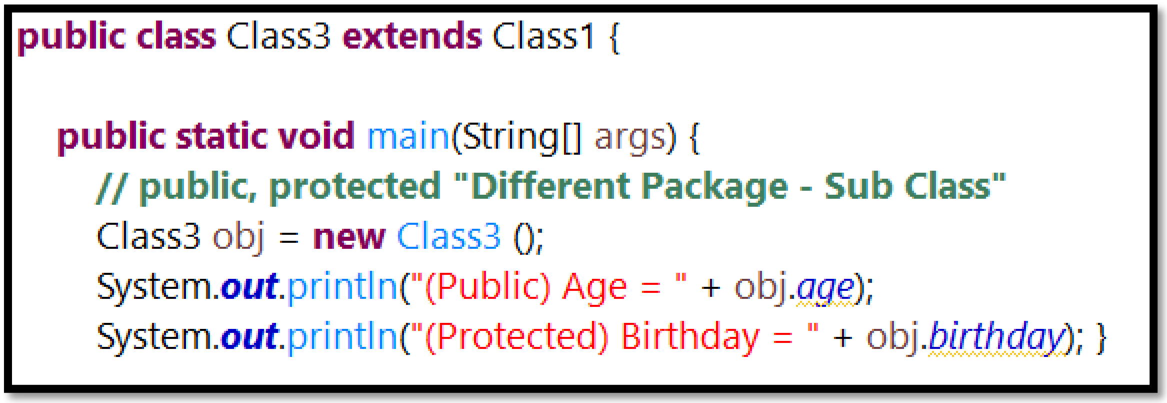
Different Package



Access Modifiers

	Public	Protected	Default	Private
Same Class	Yes	Yes	Yes	Yes
Same Package	Yes	Yes	Yes	▨
Different Package / Sub Class	Yes	Yes	▨	▨

Access modifiers get interesting when dealing with a different package. Only public and protected are visible in a different package. However, protected members are only visible if accessed through a sub class. Go to Class3 and we are going to write as a comment Different Package – Sub Class. Class3 extends Class1 and import Class1 then create an object.
Class3 obj = new Class3() / obj dot. We only see Age and Birthday which are the public and protected variables. Let's copy those print statements again but this time from Class2.

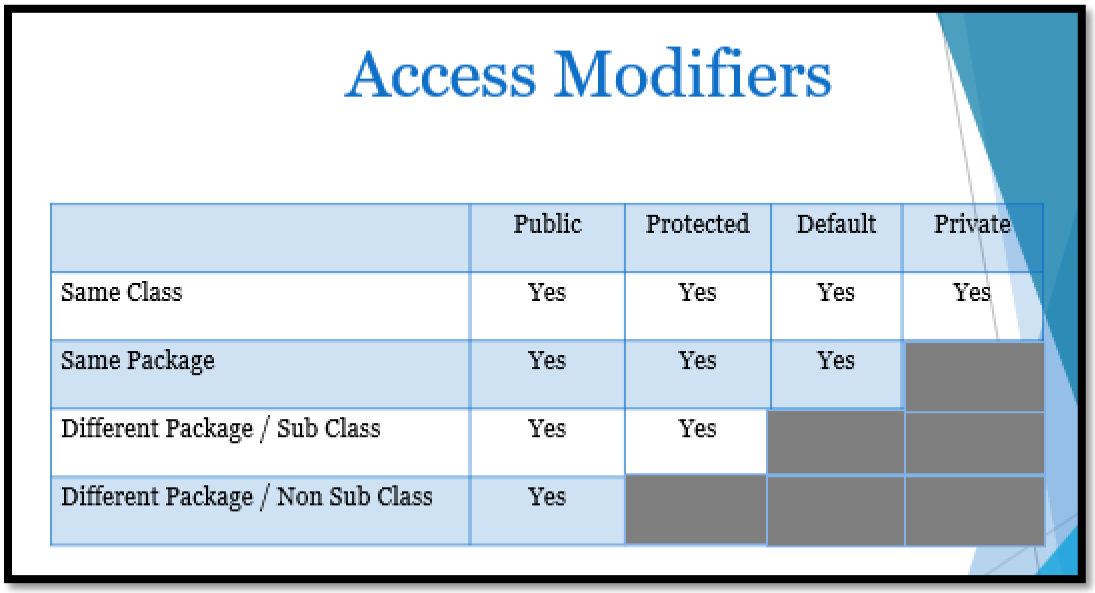


```
public class Class3 extends Class1 {

  public static void main(String[] args) {
    // public, protected "Different Package - Sub Class"
    Class3 obj = new Class3 ();
    System.out.println("(Public) Age = " + obj.age);
    System.out.println("(Protected) Birthday = " + obj.birthday); }
```

Let’s run. We see Age and Birthday

Entire Project

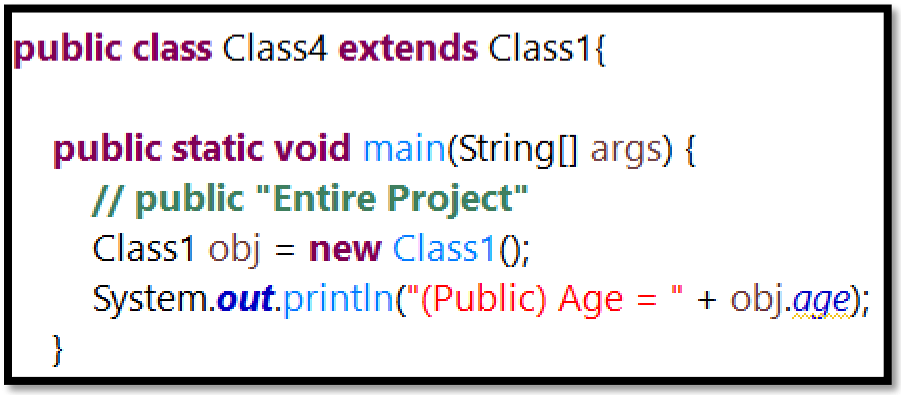


Access Modifiers

	Public	Protected	Default	Private
Same Class	Yes	Yes	Yes	Yes
Same Package	Yes	Yes	Yes	▨
Different Package / Sub Class	Yes	Yes	▨	▨
Different Package / Non Sub Class	Yes	▨	▨	▨

Only the public access modifier is visible to the entire project. That includes same class, same package, different package inside the sub class or inside super class. Go to Class4. As a comment, write “Entire Project” and extends Class1. Create an object from the super class this time: Class1 obj = new Class1() / obj dot, import Class1 and all we see is age. Print statement sysout(“(Public) Age = ” + obj.age).

Rex Jones II



```
public class Class4 extends Class1{

  public static void main(String[] args) {
    // public "Entire Project"
    Class1 obj = new Class1();
    System.out.println("(Public) Age = " + obj.age);
  }
```

Let’s run. We see Age = 20. This video applied Access Modifiers to variables but it works the same for methods. That’s it for Access Modifiers.

Subscribe to YouTube channel, connect with me on LinkedIn, and follow me on Twitter for more content.

Social Media Contact

- YouTube https://www.youtube.com/c/RexJonesII/videos
- Twitter https://twitter.com/RexJonesII
- LinkedIn https://www.linkedin.com/in/rexjones34/
- GitHub https://github.com/RexJonesII/Free-Videos
- Facebook http://facebook.com/JonesRexII