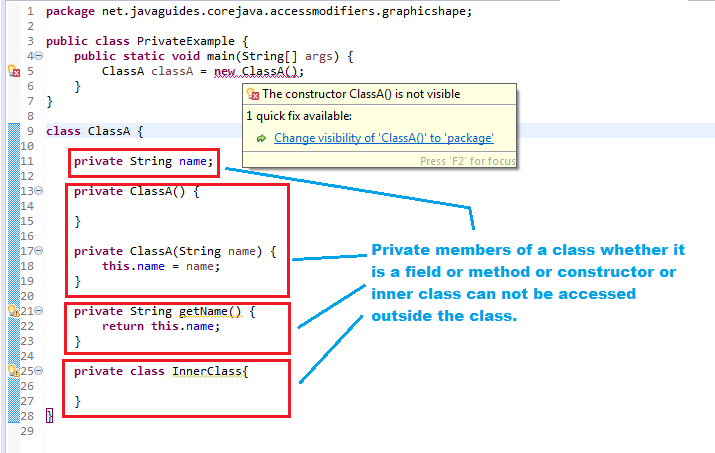
**1. Private Access Modifier**

A *private* class member cannot be accessed from outside the class; only members of the same class can access these private members.

* A class cannot be a private except inner classes because inner classes are nothing but again members of the outer class. So members of a class (field, method, constructor, and inner class) can be private but not the class itself.
* We can’t create subclasses to that class which has only private constructors.

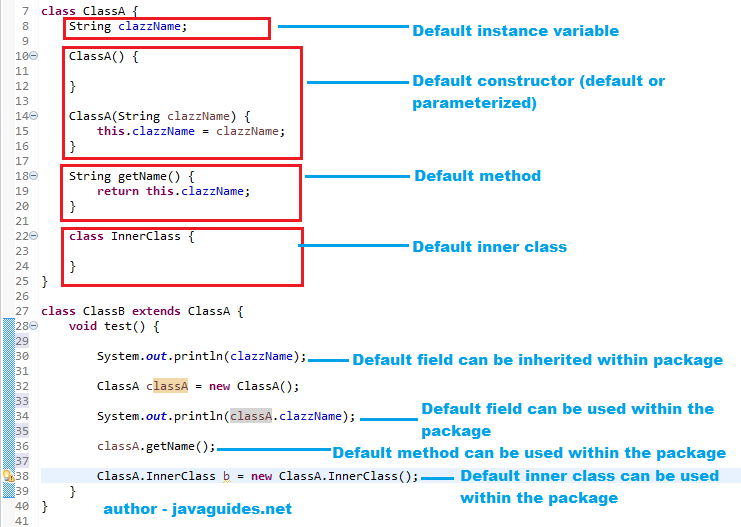
The below diagram demonstrates the usage of private access modifier and it's visibility:

****

**2. Default Access Modifier (no access modifier specified)**

When we do not mention any access modifier, it is called the **default access modifier**. The scope of this modifier is limited to the *package* only. This means that if we have a class with the default access modifier in a package, only those classes that are in this package can access this class. No other class outside this package can access this class. Similarly, if we have a default method or data member in a class, it would not be visible in the class of another package.

The below diagram demonstrates the example of *default* access modifier and it's visibility:

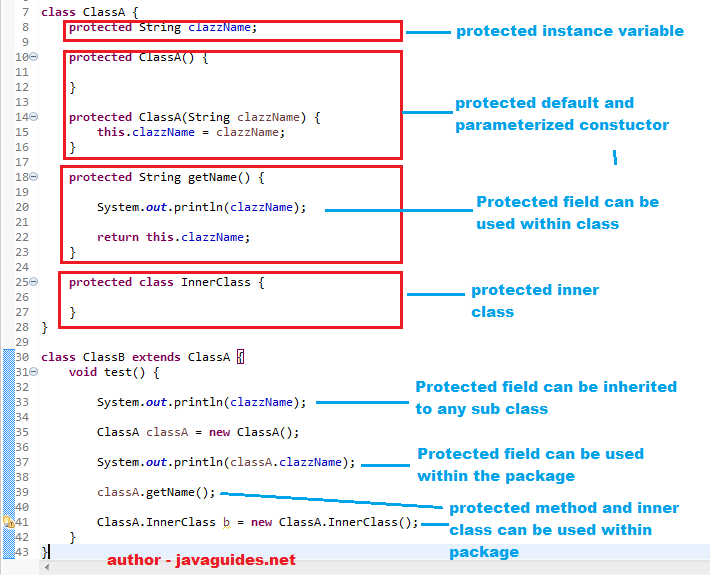
****

**3. Protected Access Modifier**

If a class or its members are declared as *protected* are only accessible by the classes of the same package and the subclasses present in any package. You can also say that the protected access modifier is similar to the default access modifier with one exception that it has visibility in subclasses.

Classes cannot be declared *protected*. This access modifier is generally used in a parent-child relationship.

The below diagram demonstrates the example of protected access modifier and it's visibility:

****

**4. Public Access Modifier**

If a class or its members are declared as *public*, they can be accessed from any other class regardless of the package boundary. It is comparable to a public place in the real world, such as a company cafeteria that all employees can use irrespective of their department.

The below example demonstrates the usage of *public* access modifiers and it's visibility. Please refer the comments are self-descriptive:

class ClassA {

public String clazzName;

public ClassA() {

}

public ClassA(String clazzName) {

this.clazzName = clazzName;

}

public String getName() {

System.out.println(clazzName); // public field can be used anywhere

InnerClass innerClass = new InnerClass(); //public Inner Class can be used anywhere.

return this.clazzName;

}

public class InnerClass {

}

}

class ClassB extends ClassA {

void test() {

System.out.println(clazzName); //public field can be inherited to any sub class

ClassA classA = new ClassA();

System.out.println(classA.clazzName); //public field can be used anywhere

classA.getName(); //public method can be used anywhere

ClassA.InnerClass b = new ClassA.InnerClass(); //public Inner Class can be used anywhere.

}

}

// below class is a different package

class ClassD extends ClassA {

void methodOfClassD() {

System.out.println(clazzName); // public field can be inherited to any sub class

ClassA a = new ClassA();

System.out.println(a.clazzName); // Public field can be used anywhere

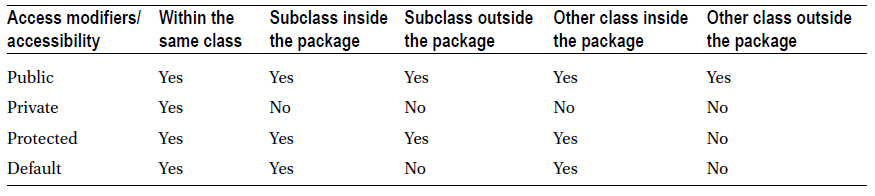
a.getName(); // Public method can be used anywhere

ClassA.InnerClass innerClass = new ClassA.InnerClass(); // Public inner class can be used anywhere

}

}

The below diagram summarized of the above concepts:

****