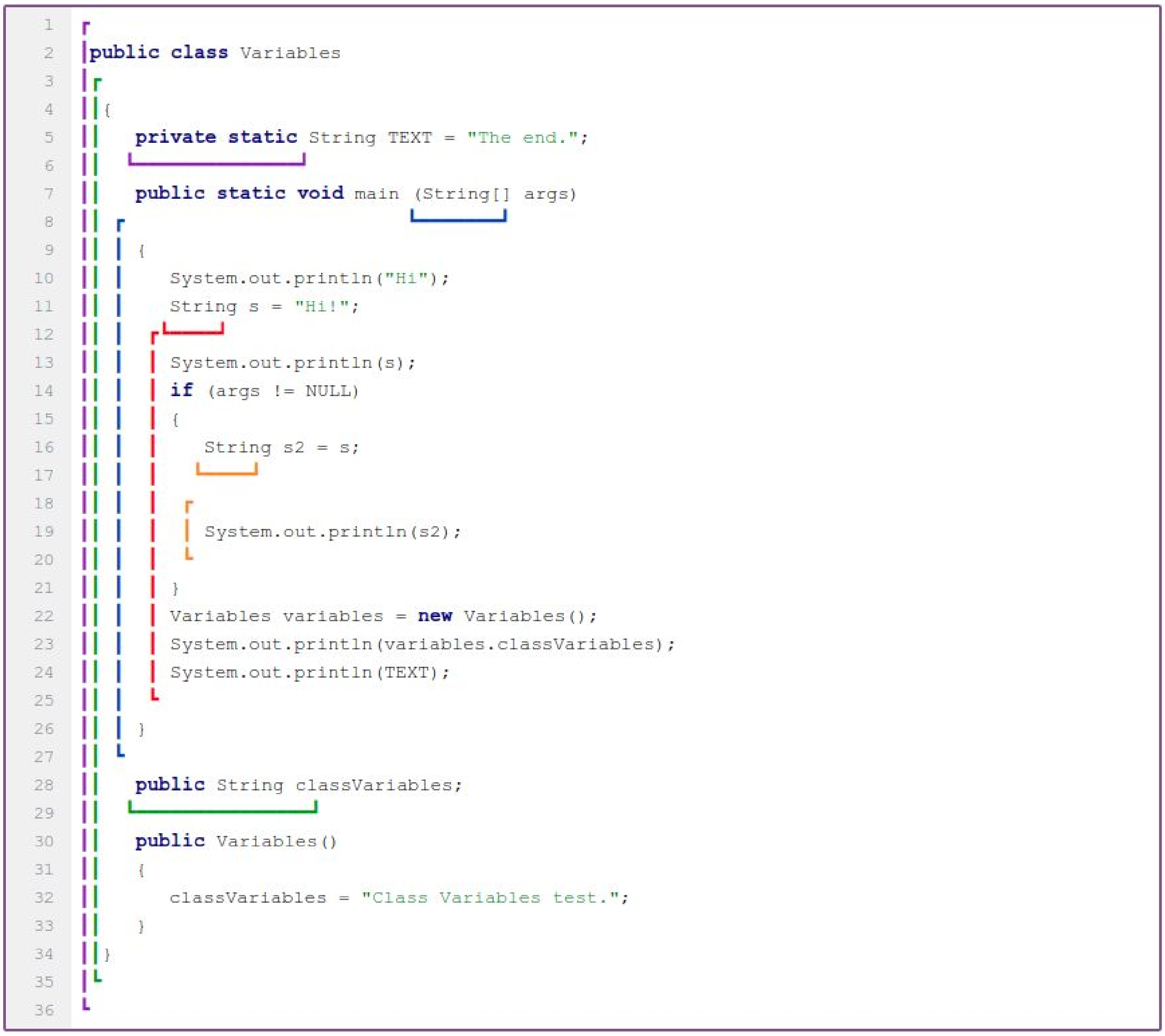
Visibility of Variables, Methods, Classes

One of the most important aspects of object-oriented design is data hiding, or encapsulation. By treating an object in some respects as a “black box” and ignoring the details of its implementation, we can write more resilient, simpler code with components that can be easily reused.



```
public class Variables

{
    private static String TEXT = "The end.";

    public static void main (String[] args)

    {
        System.out.println("Hi");
        String s = "Hi!";

        System.out.println(s);
        if (args != NULL)
        {
            String s2 = s;


            System.out.println(s2);

        }
        Variables variables = new Variables();
        System.out.println(variables.classVariables);
        System.out.println(TEXT);

    }

    public String classVariables;

    public Variables()
    {
        classVariables = "Class Variables test.";
    }
}
```

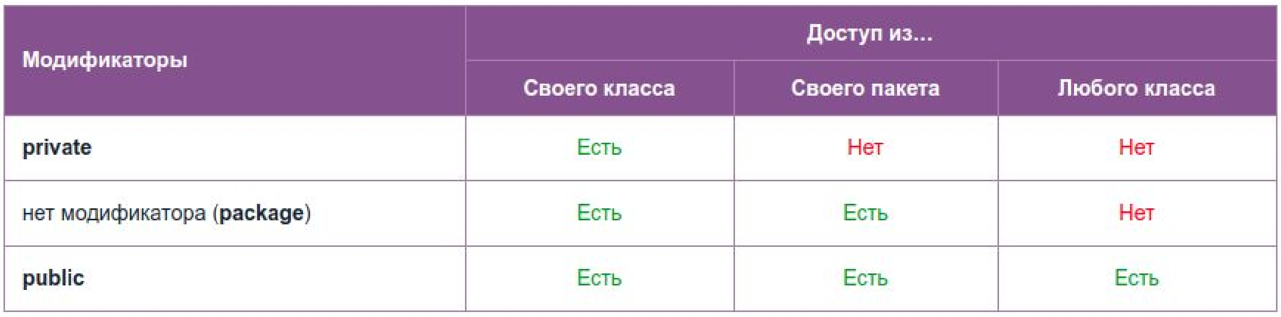
1. Переменная, объявленная в методе, существует/видна с начала объявления до конца метода.

2. Переменная, объявленная в блоке кода, существует до конца этого блока кода.

3. Переменные — аргументы метода — существуют везде внутри метода.

4. Переменные класса/объекта существуют все время жизни содержащего их объекта. Их видимость дополнительно регулируется специальными модификаторами доступа: public, private.

5. Статические переменные классов существуют все время работы программы. Их видимость также определяется модификаторами доступа.



Модификаторы	Доступ из...		
	Своего класса	Своего пакета	Любого класса
private	Есть	Нет	Нет
нет модификатора (**package**)	Есть	Есть	Нет
public	Есть	Есть	Есть

These variables that are declared inside class (outside any function). They can be directly accessed anywhere in class.

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).

```
public class Test
{
    // All variables defined directly inside a class
    // are member variables
    int a;
    private String b
    void method1() {....}
    int method2() {....}
    char c;
}
```

We can declare class variables anywhere in class, but outside methods.

Access specified of member variables doesn't effect scope of them within a class.

Member variables can be accessed outside a class with following rules

```
Modifier      Package  Subclass  World
```

public	Yes	Yes	Yes
protected	Yes	Yes	No
Default (no modifier)	Yes	No	No
private	No	No	No

Default Access Modifier - No Keyword

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.

Example

Variables and methods can be declared without any modifiers, as in the following examples −

```
String version = "1.5.1";

boolean processOrder() {
   return true;
}
```

Private Access Modifier - Private

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.

Example

The following class uses private access control −

```
public class Logger {
   private String format;

   public String getFormat() {
      return this.format;
   }

   public void setFormat(String format) {
      this.format = format;
   }
}
```

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

So, to make this variable available to the outside world, we defined two public methods: *getFormat()*, which returns the value of format, and *setFormat(String)*, which sets its value.

Public Access Modifier - Public

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.

Example

The following function uses public access control −

```
public static void main(String[] arguments) {
   // ...
}
```

The main() method of an application has to be public. Otherwise, it could not be called by a Java interpreter (such as java) to run the class.

Protected Access Modifier - Protected

Variables, methods, and constructors, which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be declared protected.

Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

Example

The following parent class uses protected access control, to allow its child class override *openSpeaker()* method −

```
class AudioPlayer {
   protected boolean openSpeaker(Speaker sp) {
      // implementation details
   }
}

class StreamingAudioPlayer extends AudioPlayer {
   boolean openSpeaker(Speaker sp) {
      // implementation details
   }
}
```

Here, if we define openSpeaker() method as private, then it would not be accessible from any other class other than *AudioPlayer*. If we define it as public, then it would become accessible to all the outside world. But our intention is to expose this method to its subclass only, that's why we have used protected modifier.

Access Control and Inheritance

The following rules for inherited methods are enforced −

- Methods declared public in a superclass also must be public in all subclasses.
- Methods declared protected in a superclass must either be protected or public in subclasses; they cannot be private.
- Methods declared private are not inherited at all, so there is no rule for them.