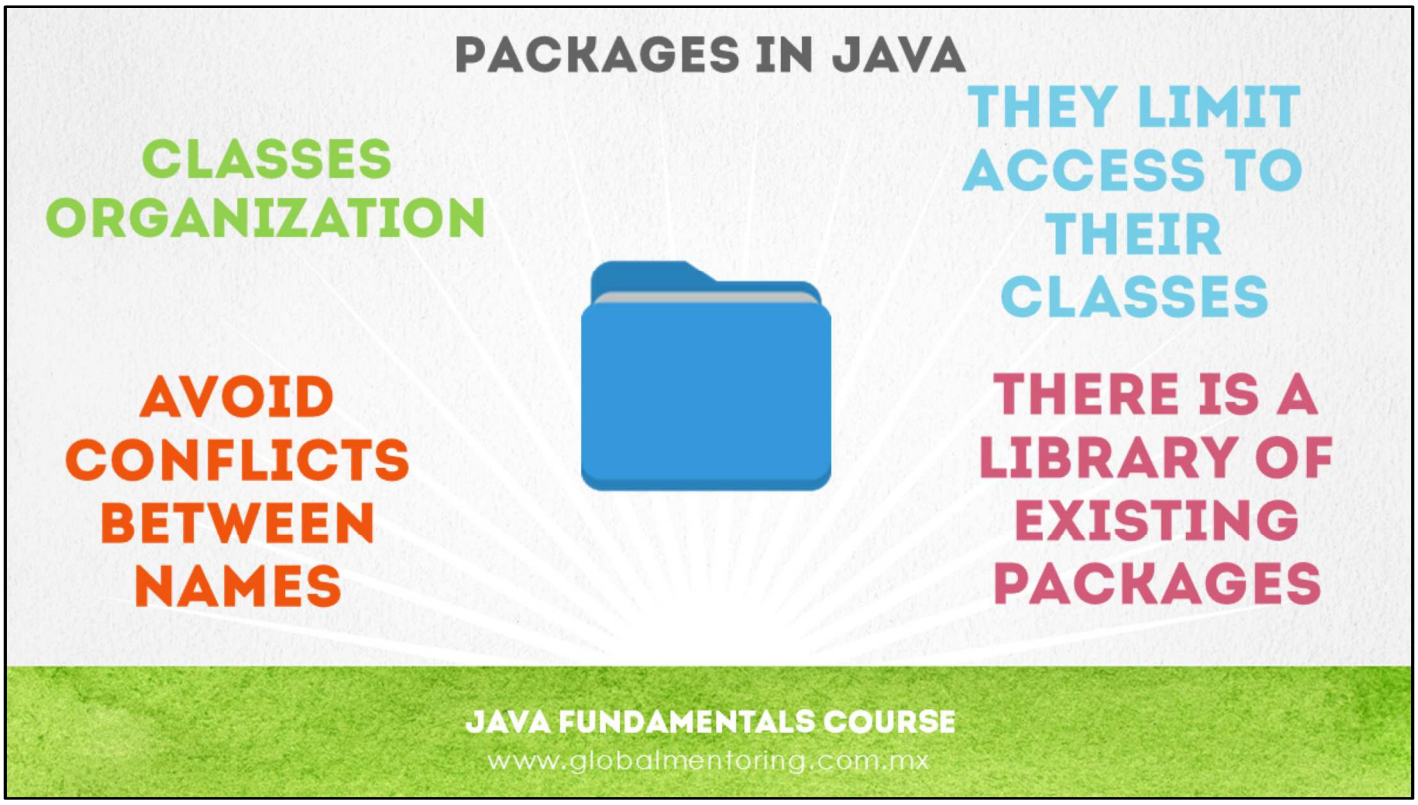
Hello, Ubaldo Acosta greets you. Welcome or welcome again. I hope you're ready to start with this lesson.

We are going to study the subject of packages in Java.

Are you ready? OK let's go!



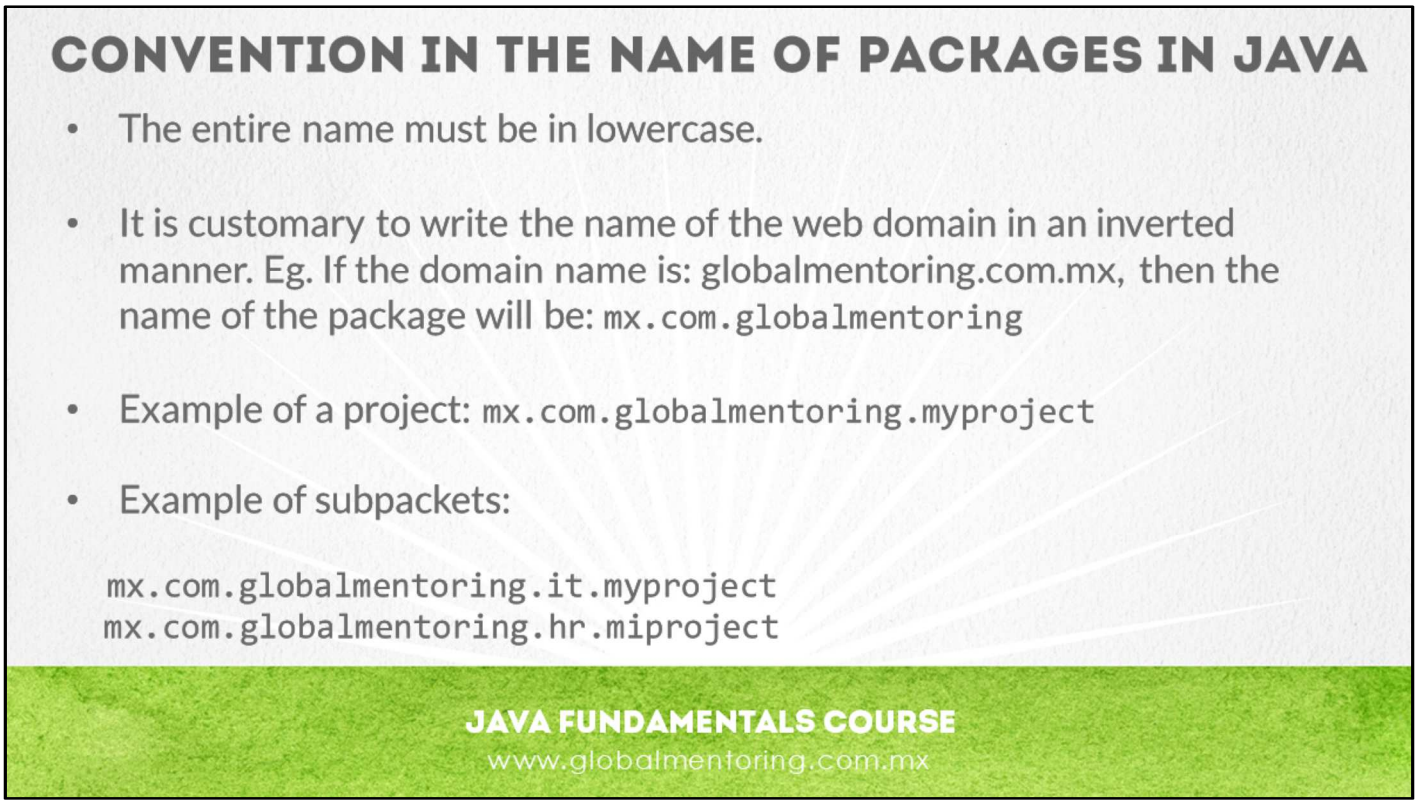
Java packages allow us to organize our classes. We can think of the packages as any folder or folder that we create to organize documents, files, photos or any type of information. In Java what we organize are classes, and basically allows us to group classes either by function, by inheritance, or by any feature that we want, the important thing is that we have an organization of our classes, since only in the standard version of Java We find more than 4000 classes, and therefore it is important to define the way we organize our classes.

In the same way, a project in Java usually has hundreds or thousands of classes, therefore we must organize our classes in the best way, usually because of the function that these classes play in our program.

Having so many classes, it is normal that there are classes with the same name, however as each class belongs to a different package, the packages will also help us to avoid name problems between classes. For example there are many Date classes in the standard version of Java, and to be able to distinguish which class we want to use we must specify the Java package where this class is located. Later we will see an example of how to use the packages, and how Java allows us to use the classes that belong to those packages.

Another use of the packages is that they limit access to the methods or attributes of our classes, since it is limited according to the access modifiers that are used. We will see this concept in detail in the next course.

In the standard version of Java there are more than 4000 classes, so there is already a large number of defined packages, in addition to those that we define. All of these classes are available for use in our Java programs, and we will see later how we can access these classes.



The name of the packages in Java follow a standard.

The name of the domain does not consider www, it is only the name of the domain. This ensures that this name used on the internet is unique. If we did not have a domain, we can assign a name to the project and imaginatively create our internet name. For example, if you have a name conflict with another project, simply change the name of the project to someone who does not have a conflict. The idea is that we follow this standard even if we do not have a domain on the internet, since this web domain is not necessary to exist on the internet, it is only to avoid name conflicts as much as possible.

However, we can see in the slide an example of the domain globalmentoring.com.mx, which if we want to use it in our Java projects we could use it, first by inverting the order of the domain name, and then adding the name of our project, with the goal of having a unique package name. We also see some examples if we had a project, and if that project had subpackets, for different parts of the same project.

EXAMPLE OF CLASS INSIDE PACKAGE

Example of class inside package in Java:

```
package com.gm; //Package definition

public class Utility{
    public static void print(String s){
        System.out.println("Printing message: " + s);
    }
}
```

```
import com.gm.*; //Import the package that will be used

public class PackageExample {

    public static void main(String[] args) {
        Utility.print("Hello"); //We use the imported class
    }
}
```

We can see in the example that we first created a class called Utility, this class is adding it to a package called com.gm, but it can be any name they want.

Once we have added one or more classes to this package, then we create a class in a different package. In this case we create the PackageExample class in the Java default package, that is, in no package. It should be noted that this practice is not recommended, it is only to perform the exercise. Java recommends that all classes that we create must be in a package.

Now, to be able to use the Utility class defined in the com.gm package, what we should do is use the word import, which can be used in two ways, one is importing all the classes using the *, or the other is specifying the name of the class, that is: import com.gm.Utility. However, this last option forces us to make an import for each class that we want to use from the com.gm package, and if there were many classes it would be many lines of code, so in many cases the notation import com.gm. * is the most used. It should be mentioned that the import does not affect the memory, a class loads in memory until the name of the class is found within the program and not before.

Finally we can see how we are using the static method called print, which belongs to the Utility class. We note that we are no longer indicating which package corresponds to that class, since we have done the import. However, if there is more than one class called Utility, we could eliminate the import and indicate in the line of code where we are going to use the class, directly the package to which the class corresponds, being com.gm.Utility.print("Hello"); with this we would know exactly to which package corresponds the class that we are using, nevertheless this is not a common practice.

STATIC IMPORT

Static import example in Java:

```java
package com.gm; //Package definition

public class Utility {
    public static void print(String s){
        System.out.println("Print message: " + s);
    }
}
```

```java
//Import the static method to use inside this class
import static com.gm.Utility.print;

public class PackageExample {

    public static void main(String[] args) {
        print("Hello");
    }
}
```

In Java it is possible to import the static methods to be used, in this way we simplify the syntax in the static methods within our code.

In the example shown we can see that the Utility class and the package they belong to are the same as in the previous example, but because the print method is a static method, we can take advantage of the static import syntax provided by java to import the static method to be used, and in this way the syntax of the use of the print method is simplified, since as we can see the print method, it should no longer indicate which class it belongs to, but it is sufficient to indicate the name of the method, since in the static import has already been indicated that it belongs to the Utility class.

MOST IMPORTANT JAVA SE PACKAGES

Package Name	Content
java.lang	It contains essential classes, it is implicitly imported without the need for an import sentence
java.util	Contains classes such as the Date class, among many kinds of common tools in Java
java.io	input/output. Classes that define different data flows
java.net	It is used in combination with the classes in the java.io package to read and / or write data on the network.
java.applet	Contains the classes needed to create applets and run in the browser window
java.awt	Contains classes to create a platform-independent GUI (Graphic User Interface) application

www.globalmentoring.com.mx

In the table shown, we can see some of the most common Java packages of the standard version. The most important is the java.lang package, which contains the core classes of the Java language, such as the Object class, which is the class from which all Java classes descend, among several other classes. It should be noted that it is not necessary to import any of the classes in this package, since it is the only one that is imported automatically because of the importance of these classes in the compilation and execution of our Java classes.

There are other packages which are also very important, such as the package java.util, java.net, and java.awt, java.ui and java.applet, among many more packages. As we progress through the course and later courses we will be studying several of the classes of these packages.

If you want to know all the classes that exist in the Java Standard Edition, every package, class, and it's methods, besides many more elements, you can check the JavaDoc of the version you are interested. For example:

https://docs.oracle.com/javase/8/docs/api/