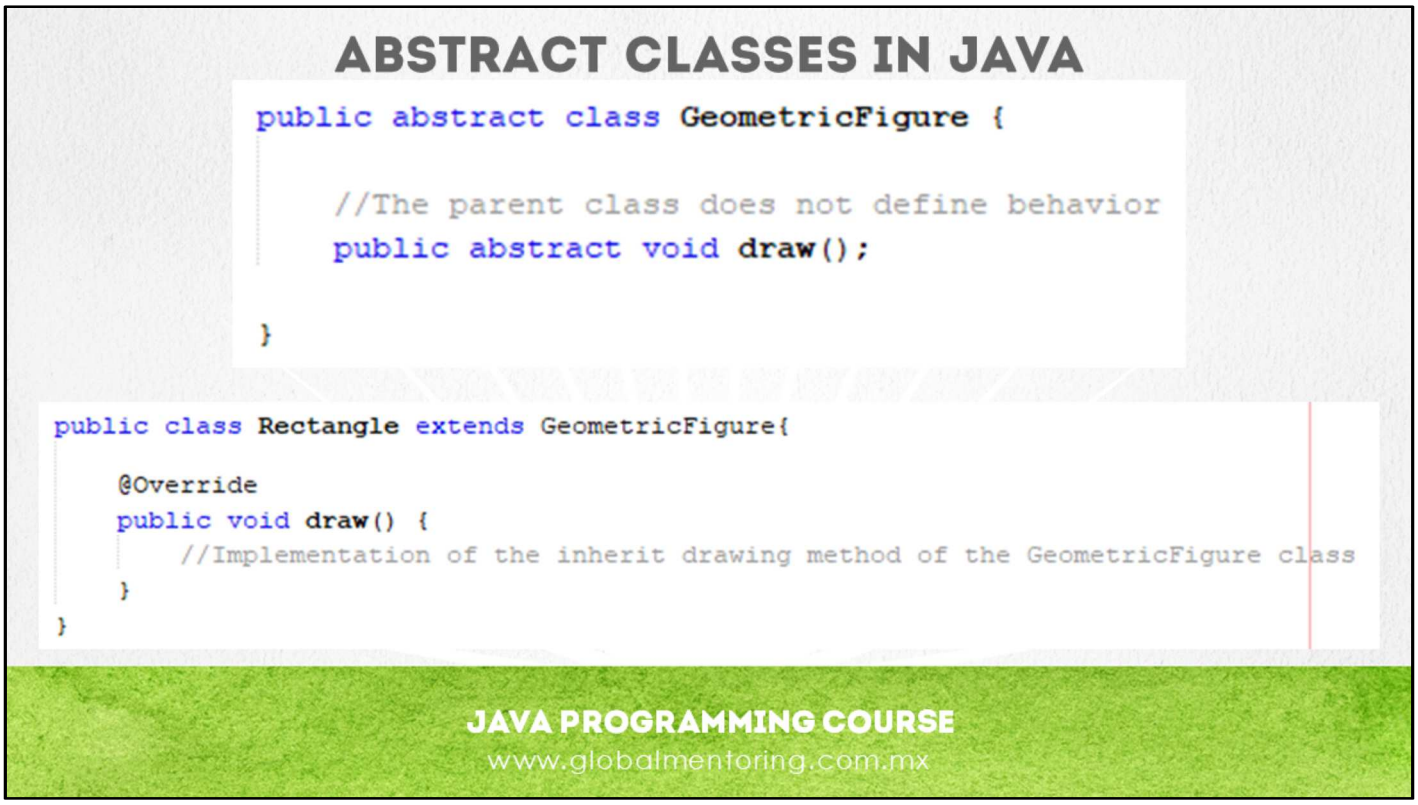
Hello, Ubaldo Acosta greets you. Welcome or welcome again. I hope you're ready to start with this lesson.

We are going to study the subject of abstract classes in Java.

Are you ready? OK let's go!

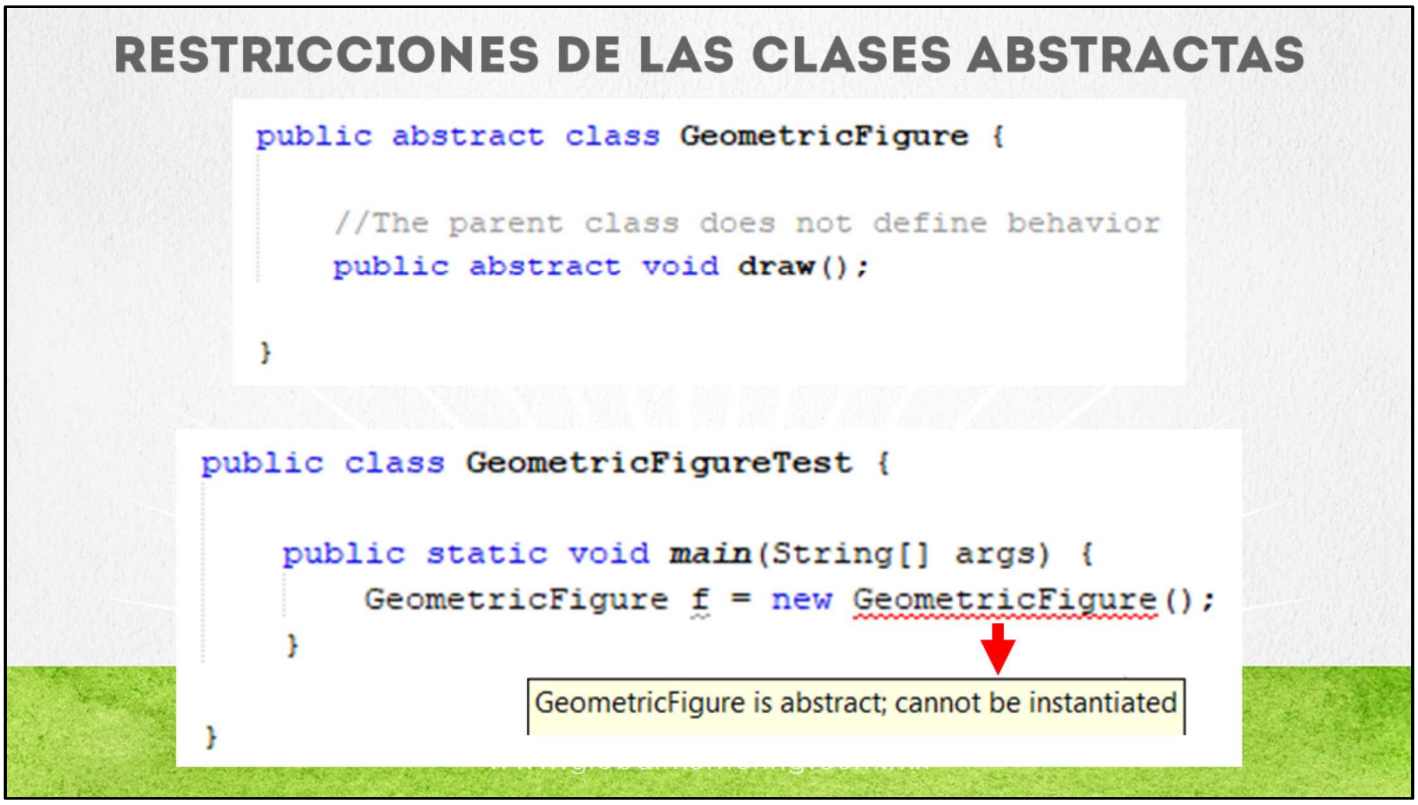


In this lesson we are going to study the subject of abstract classes. In the example shown, the draw method of the parent class (GeometricFigure), it does not make sense to draw, because you have to specify what type of geometric figure is to be drawn, and this we will know in some of the subclasses, such as Rectangle, Triangle, Circle, etc.

The Rectangle subclass is the class responsible for implementing the functionality of the draw method inherited from the parent class, and then it makes sense to add a behavior to the draw method.

If a class contains an abstract method, the class must also be declared as abstract, this is a Java compiler rule.

An abstract class can contain abstract and non-abstract methods, in addition an abstract method has no body, and only ends with a semicolon, in this way an abstract method only defines the signature of the method but does not add any behavior, and it is the child classes those who are responsible for implementing this method, that is, add some behavior. In some ways it is like overwriting, however, because in the parent method no behavior has been defined in the abstract method, then it is not said that the daughter class overwrites the legacy method, but implements it.



An abstract class can not be instantiated, this is because something abstract can not materialize in an object, and this same concept applies to abstract classes in Java.

What we can do is create type variables of the abstract class that store references to subclasses, and thus call the methods in common between the parent class and the subclass, for example the draw method. This is very similar to the polymorphism we studied earlier. If we assign a reference of a Rectangle object (subclass) to a variable of type GeometricFigure (parent abstract classes), then it applies the same donut diagram that we studied in the polymorphism lesson, and therefore the method that will be executed will be the method implemented by the subclass, that is, the draw method defined by the Rectangle class.

So the same rules of polymorphism apply in the case of abstract classes and therefore we can continue to take advantage of more generic types, such as a type of abstract class and assign object references of child classes and thus create more generic methods that use the concept of polymorphism.