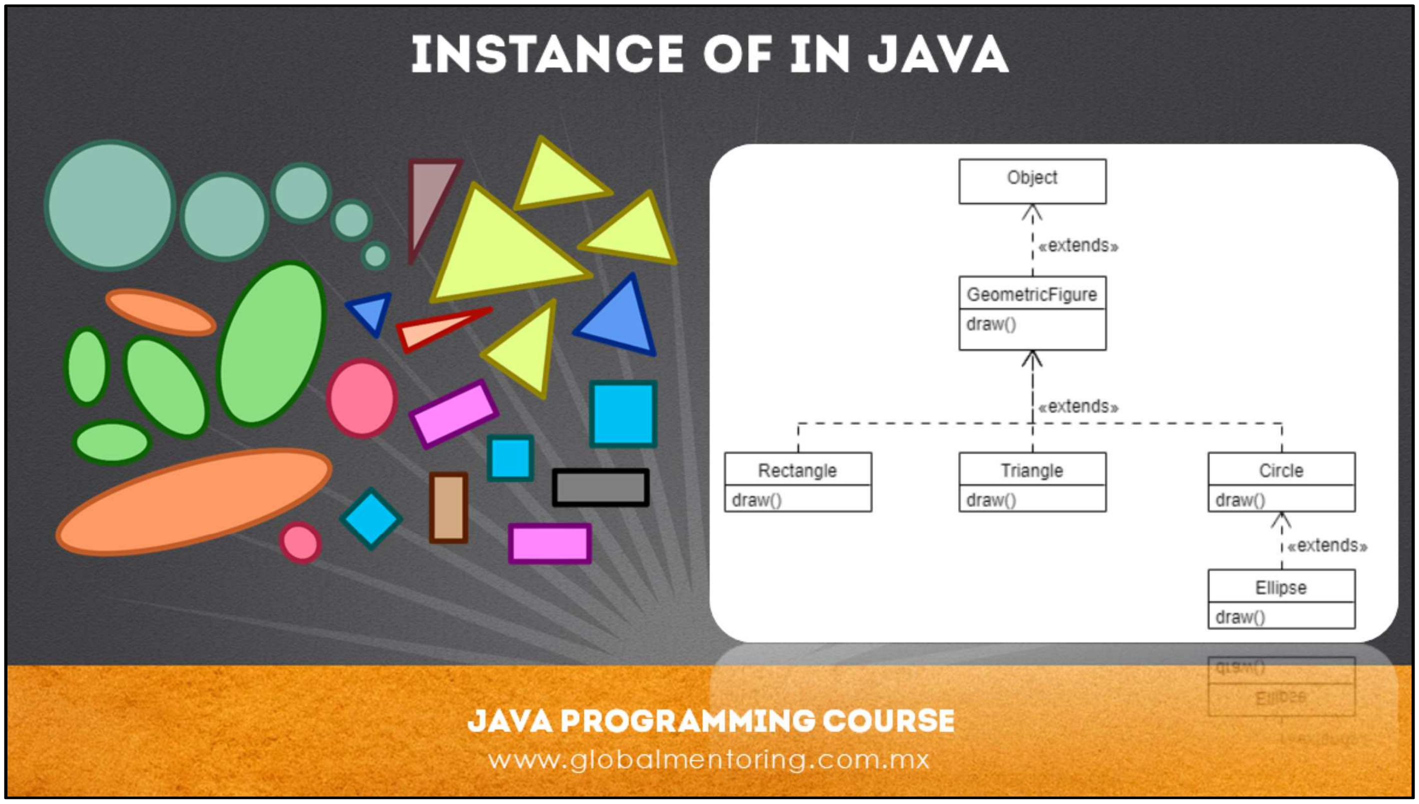
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson ..

We are going to study the topic instance of in Java.

Are you ready? Come on!

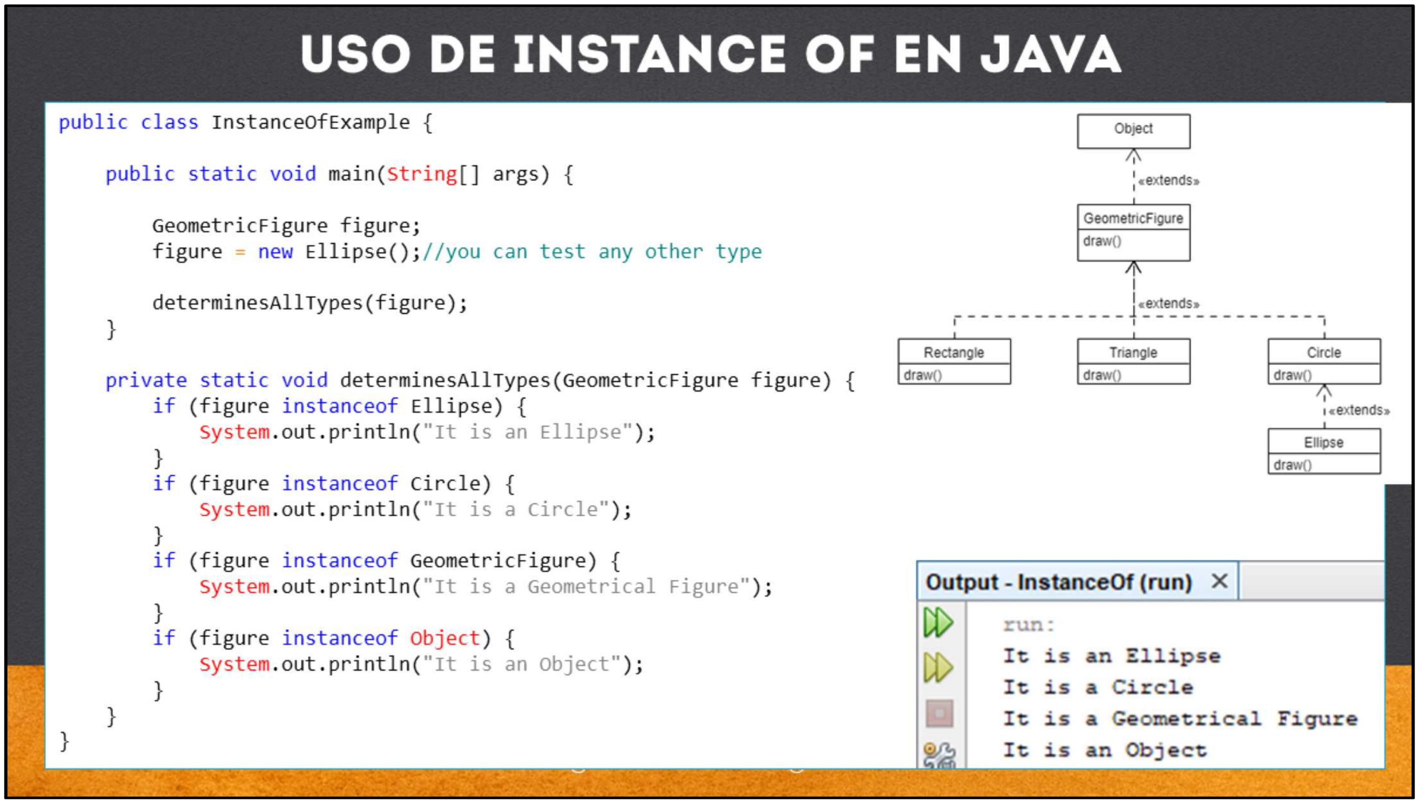


The instaceof keyword basically answers the question if a class has any relationship with any other class that we indicate.

For example in the figure, we can see that all the elements could be part of a class called Geometric Figure, that is, this would be the super class or parent class of all geometric figures shown. And as subclasses we would have the class Circle, Triangle, Rectangle, Ellipses, etc.

And in turn each of these classes would in turn be a type of geometric figure. Therefore, if we asked if a circle is a geometric figure, the answer should be yes, similarly if we asked this question of a rectangle or a triangle. To ask this question, there is precisely the reserved instanceof keyword, which means is it a type of this other indicated type?

We also see a class diagram with which we can add a class hierarchy and thus be able to ask the question if a class is of a certain type, and we see that if there is a hierarchy of classes involved then the logical thing should be that they are of the type which they inherit. Let's see an example based on this class diagram, how to ask this question in Java code and what is the use of asking this question.



USO DE INSTANCE OF EN JAVA

```java
public class InstanceOfExample {

    public static void main(String[] args) {

        GeometricFigure figure;
        figure = new Ellipse();//you can test any other type

        determinesAllTypes(figure);
    }

    private static void determinesAllTypes(GeometricFigure figure) {
        if (figure instanceof Ellipse) {
            System.out.println("It is an Ellipse");
        }
        if (figure instanceof Circle) {
            System.out.println("It is a Circle");
        }
        if (figure instanceof GeometricFigure) {
            System.out.println("It is a Geometrical Figure");
        }
        if (figure instanceof Object) {
            System.out.println("It is an Object");
        }
    }
}
```

```
run:
It is an Ellipse
It is a Circle
It is a Geometrical Figure
It is an Object
```

The "instanceof" operator is used to know if an object is an instance of a specific class at runtime. If it corresponds with the data type returns "true", otherwise "false".

As we can see in the code, the Ellipse type due to the class hierarchy that we have defined to answer that IF is of the type for several types, for example, it responds that if it is of type Ellipse, Circle and GeometricFigure, even if we ask if it is of type Object the answer will be true, because all the classes indirectly inherit in the first term of the Object class.

Therefore, if we are going to make a revision of several types in the class hierarchy, it is recommended that the lowest hierarchy classes be reviewed first and the highest hierarchy classes, because a class could result in " true "if it is a compatible type, which can be promoted to a data type. This promotion concept we will see later, but basically if we ask first for classes of higher hierarchy we return true, and cases of lower classes in the class hierarchy could no longer be reviewed.

If we want it to only coincide with the most internal type, instead of using the if structure, we can use the if-else structure, with which it will only enter a case when checking if it is of a certain type. It is worth mentioning that the instance of operator can only be used in variables of type Object, but not in primitive types.

Now, what is the use of knowing if a type is of a certain type in a hierarchy of classes? And the answer is that depending on the type that is a class, we can perform certain operations for being of a certain type or send to call some method for a certain type, and this we would only do if it responds as true to a data type, otherwise we would not execute that code. So it's a way to start creating more complex code depending on the class hierarchy we have. In some way it is like manually programming the polymorphism, but here it is not limited to execute a method of a certain type, but we can execute any code, or send to call several methods, etc., so the instanceof operator allows us to know the type of data of the object at run time and thus be able to perform some action due to the type that we have detected.

ONLINE COURSE

JAVA PROGRAMMING

By: Eng. Ubaldo Acosta

Global Mentoring

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx