# Static and Instance in Apex Salesforce

**Comparte la información a tus amigos**

If you are a Salesforce developer, you need to know everything about static and instance in Apex Salesforce to code as a pro with good programming principles.

In apex classes can't be static. Nonetheless, we can have static methods, variables and initialization code. Also we can have instance methods, member variables and initialization code, which have no modifier, and local variables.

## Static Characteristics in Apex

Static methods, variables and initialization code could be:

- Associated with a class.
- Only in **outer classes**.
- Only **when** a **class is loaded**.
- **No transmitted** as part of the **view state** for a **Visualforce page**.

## Instance Characteristics in Apex

Instance methods, member variables, and initialization code could be:

- Associated with a **particular object**.
- No definition modifier.
- **Created** with **every object instantied** from the class declared.

## Using Static Methods and Variables

- Only with **outer classes**. That means, classes that can have inner classes.
- **Doesn't require** an **instance** of the class in order to run.
- All **static variables** are initialized **before** an **object** of a class is **created**.
- A **static method** is used as a utility method. For example, when we need to access to raw data without make an object declaration.
- A **static variable** is static **only within** the **scope** of the Apex **transaction**. Can be used as a **flags** in Apex **Triggers**.
- A static variable or method **can't be accessed through** an **instance** of that class.
- An **inner class behaves** like a **static class** and doesn't require the **static** keyword. Those classes doesn't need to be declared.
- In other words, **use static** methods and variables when you **don't need** the abstraction of **an object** but straight code.
- *Ask yourself "Does it **make sense to call this method**, even **if no object has been constructed yet**?" **If so**, it should definitely be **static**.*

```
public class P {
   public static boolean firstRun = true;
}
```

```
trigger T1 on Account (before delete, after delete, after undelete) {
  if(Trigger.isBefore){
    if(Trigger.isDelete){
        if(p.firstRun){
            Trigger.old[0].addError('Before Account Delete Error');
            p.firstRun=false;
        }
      }
  }
}
```

## Using Instance Methods and Variables

- Are **used** by instance of a class, that is, **by an object**.
- Instance member variable is **declared inside** a **class**.
- Usually use instance member variables **to affect the behavior of the method**.
- In other words, use it **when depends of an object** and could **affect** their **behaviour**. For example, a **Point**, need an **instance** of **x** and **y** to be created.

```java
public class Plotter {

    // This inner class manages the points
    class Point {
        Double x;
        Double y;

        Point(Double x, Double y) {
             this.x = x;
             this.y = y;
        }
        Double getXCoordinate() {
             return x;
        }

        Double getYCoordinate() {
             return y;
        }
    }

    List<Point> points = new List<Point>();

    public void plot(Double x, Double y) {
        points.add(new Point(x, y));
    }

    // The following method takes the list of points and does something with them
    public void render() {
    }
}
```