Local Class ve Alt Class

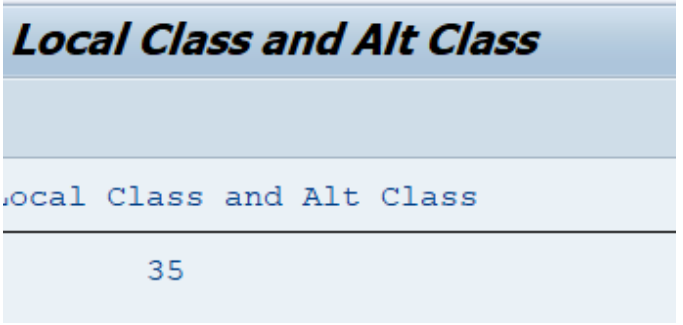
Global class her yerde kullanılabilen bir yapı, Local ise oluşturulduğu objenin altında kullanılmasını sağlayan yapı. Local olarak oluşturulan bir class sadece o program/obje altında kullanılabilir.

Class oluştururken iki mantık vardır, birincisi implementasyon ikincisi ise definition

Definition kısmında, class içinde kullanılacak dataları, methodları tanımlarken. İmplementasyon kodlamasını yapıyoruz.

```
CLASS math_op DEFINITION. "Bu kısımda datalar, methodlar vb.. tanımlamalar yapılır.
  PUBLIC SECTION.
    DATA: lv_num1   TYPE i,
          lv_num2   TYPE i,
          lv_result TYPE i.

    METHODS: sum_numbers.
ENDCLASS.

CLASS math_op IMPLEMENTATION. "Bu alan ise kodlama blok alanıdır.
  METHOD sum_numbers.
    lv_result = lv_num1 + lv_num2.
  ENDMETHOD.
ENDCLASS.

DATA: go_math_op TYPE REF TO math_op.

START-OF-SELECTION.
  CREATE OBJECT: go_math_op.

  go_math_op->lv_num1 = 12.
  go_math_op->lv_num2 = 23.
  go_math_op->sum_numbers( ).

  WRITE: go_math_op->lv_result.
```

Görüldüğü gibi Definition içerisine tanımları yaptıktan sonra Implementation içerisinde ise bunların kullanımına uygun şekilde nasıl yapılacağını görmüş ve göstermiş olduk. Sonuç olarak ekrana 35 yazması gerekiyor, sonuç olarak;



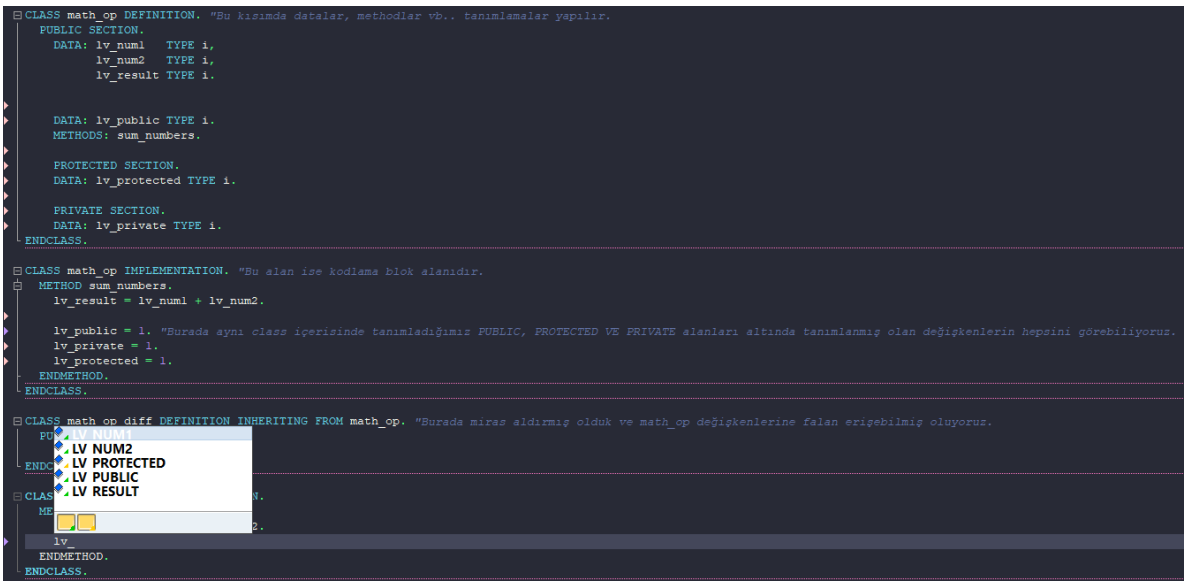
Bunun ardından gelecek olan ise Alt Class, yani inherit edilmiş miras almış class. Burada bizim bir classımız olan go_math_op_diff mevcut ve biz buna ilk classtan değişkenleri ve methodları dahil etmeyi istiyoruz, işte bu nedenle miras aldırıyoruz. O da "Inheriting From" sayesinde yapılabiliyor.

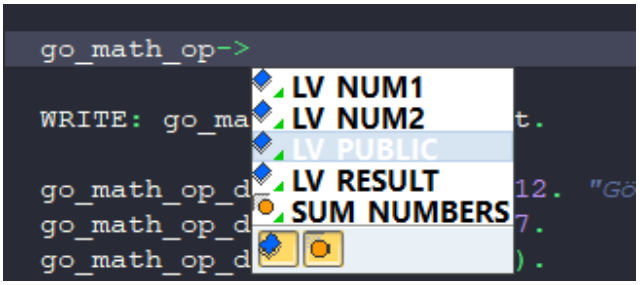
```
CLASS math_op_diff DEFINITION INHERITING FROM math_op. "Burada miras aldırmış olduk ve math_op değişkenlerine falan erişebilmiş oluyoruz.
  PUBLIC SECTION.
    METHODS num_diff.
ENDCLASS.

CLASS math_op_diff IMPLEMENTATION.
  METHOD num_diff.
    lv_result = lv_num1 - lv_num2.
  ENDMETHOD.
ENDCLASS.
```

```
DATA: go_math_op TYPE REF TO math_op.
DATA: go_math_op_diff TYPE REF TO math_op_diff.

START-OF-SELECTION.
  CREATE OBJECT: go_math_op.
  CREATE OBJECT: go_math_op_diff.

  go_math_op->lv_num1 = 12.
  go_math_op->lv_num2 = 23.
  go_math_op->sum_numbers( ).

  WRITE: go_math_op->lv_result.

  go_math_op_diff->lv_num1 = 12.
  go_math_op_diff->lv_num2 = 7.
  go_math_op_diff->num_diff( ).

  WRITE: go_math_op_diff->lv_result.
```

Visible for	PUBLIC SECTION	PROTECTED SECTION	PRIVATE SECTION
Same class and its friends	X	X	X
Any subclasses	X	X	-
Any repository objects	X	-	-

ENCAPSULATION





Visible for	PUBLIC SECTION	PROTECTED SECTION	PRIVATE SECTION
Same class and its friends	X	X	X
Any subclasses	X	X	-
Any repository objects	X	-	-

Bu encapsulation konusunda ise önemli bir detay şu; ana class içerisinde tanımlanmış olan public, protected ve private kendi içerisinde her yerde kullanılabilir durumdadır.

Fakat miras aldırılmış olan bir class içerisinde bunlara ulaşmaya çalıştığımız zaman private kısmı bize gözükmez, sadece public ve protected tanımlayabiliyoruz oluruz. Bunun haricinde eger program disinda ikincil class olan, miras almış class içerisinden protected'a ulaşmaya çalıştığımızda bile protected bize gözükmeyen bir durumda olur böylece sadece public değişkene ulaşabiliyor oluruz.