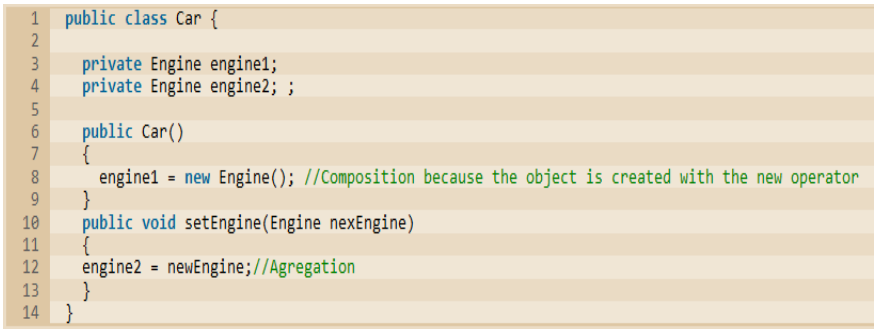
Modul 226			
Handlungsziel	A	B	C

TITEL	**Aggregation Komposition**
BESCHREIBUNG	Beziehung: Teile und Ganzes Komposition: Teil ist existenzabhängig vom Ganzen (enthält) Aggregation: Teil kann unabhängig weiter existieren (hat ein)
IDEE, BEISPIEL	
SCHWIERIGKEIT	UML macht Unterscheidung, in der Realisation vielfach kein Unterschied
SKALIERBARKEIT	Zusammenstellung eines Fussballteams; Spieler mutieren; Menu Steuerung zur wahlweisen Zuordnung von einzelnen Elementen
VORGEHEN	1. Beschreibung, 2. Pflichtenheft, 3. AD/ZD

```
1  public class Car {
2
3    private Engine engine1;
4    private Engine engine2; ;
5
6    public Car()
7    {
8      engine1 = new Engine(); //Composition because the object is created with the new operator
9    }
10   public void setEngine(Engine nexEngine)
11   {
12   engine2 = newEngine;//Agregation
13   }
14 }
```

Aggregation ['HAS-A' Relationship]

```
class StereoSystem {
    private boolean state ;
    StereoSystem() {}
    StereoSystem(boolean state) {
        this.state = state ;
        System.out.println("Stereo System State: " + (state == true ? "On!" : "Off!")) ;
    }
}
class Car {
    private StereoSystem s ;
    Car() {}
    Car(String name, StereoSystem s) {
        this.s = s ;
    }
    public static void main(String[] args) {
        StereoSystem ss = new StereoSystem(true) ;  //  true(System is ON.)   or  false (System is OFF)
        Car c = new Car("BMW", ss) ;
    }
}
```

In UML Aggregation is Expressed as

Composition ['Contains' Relationship]

```
1  import java.util.Date ;
2  class Piston {
3      private Date pistonDate ;
4      Piston() {
5          pistonDate = new Date() ;
6          System.out.println("Manufactured Date :: " + pistonDate) ;
7      }
8  }
9  class Engine {
10     private Piston piston ;
11     Engine() {
12         piston = new Piston() ;
13     }
14     public static void main(String[] args) {
15         Engine engine = new Engine() ;
16     }
17 }
```

In UML Composition is Expressed as