Κλάσεις και Αντικείμενα

Η εφαρμογή «Το αυτοκίνητο»

Σε μια εφαρμογή χρειάζεται να διαχειρίζομαι πολλά και διαφορετικά αυτοκίνητα. Αποφασίζω να περιγράψω ένα αυτοκίνητο από τα βασικά του στοιχεία, την εταιρία κατασκευής, το μοντέλο, το έτος κατασκευής. Για παράδειγμα, το μοντέλο είναι **Zafira** της εταιρίας **Opel** με έτος κατασκευής το **2008**.

Από το είδος των στοιχείων που χρειάζονται για να περιγράψω ένα αυτοκίνητο (μοντέλο, εταιρία, έτος) αποφασίζω να επιλέξω σαν τύπους δεδομένων, τα παρακάτω :

μοντέλο : Αλφαριθμητικό (String)
εταιρία : Αλφαριθμητικό (String)
Έτος : Ακέραιος αριθμός (int)

Δημιουργία εφαρμογής για ένα αυτοκίνητο

```
public class Example {

   public static void main(String[] args) {
      String company, model;
      int prodyear;

      company = "Porsche";
      model = "911";
      prodyear = 1975;
      System.out.println(company + " " + model + " " +prodyear);

   }
}
```

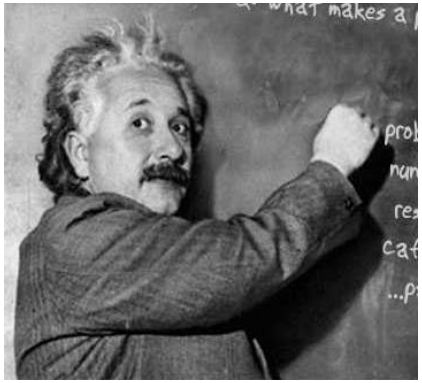
Τι προβλήματα εντοπίζεται στον ανωτέρω κώδικα;

Δημιουργία εφαρμογής για δύο αυτοκίνητα

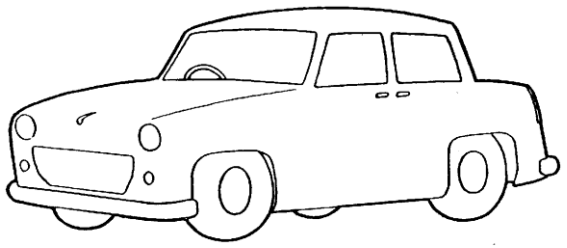
```
public class Example {

   public static void main(String[] args) {
      String company01, model01;
      int prodyear01;

      company01 = "Porsche";
      model01 = "911";
      prodyear01 = 1975;
      System.out.println(company01 + " " + model01 + " " + prodyear01);

      String company02, model02;
      int prodyear02;

      company02 = "Toyota";
      model02 = "Hilux";
      prodyear02 = 1988;
      System.out.println(company02 + " " + model02 + " " + prodyear02);
   }
}
```

Τι προβλήματα εντοπίζεται στον ανωτέρω κώδικα;

Η κλάση αυτοκίνητο



```
public class Car {



}
```



Θα δημιουργήσουμε τη **κλάση** αυτοκίνητο, δηλαδή ένα σχέδιο που περιγράφει γενικά ένα αυτοκίνητο, στο οποίο θα δηλώσουμε με ακρίβεια τα δεδομένα που απαιτούνται για τη περιγραφή του – ονομάζονται **ιδιότητες**, τις ενέργειες που μπορούμε να κάνουμε – ονομάζονται **μέθοδοι**, τον τρόπο κατασκευής του – ονομάζονται **κατασκευαστές**, το είδος της πρόσβασης στα δεδομένα και τις ενέργειες κ.λπ.

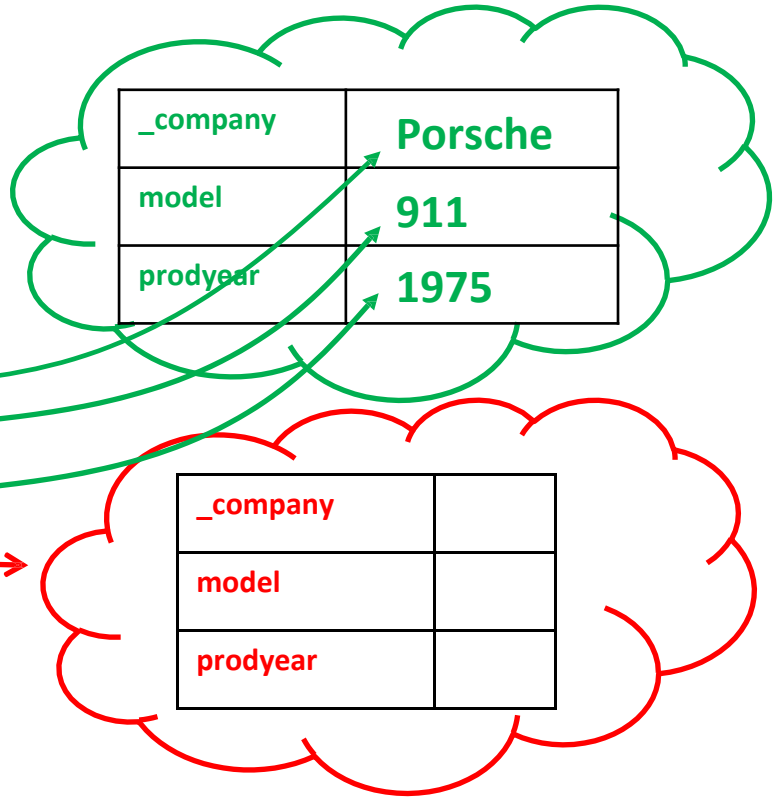
Εν συνεχεία, βάση του σχεδίου θα κατασκευάσουμε **αντικείμενα της κλάσης**, τα οποία όλα περιέχουν την ίδια λειτουργικότητα αφού προέρχονται από το ίδιο σχέδιο, μόνο που το καθένα έχει τις δικές του τιμές.

Η κλάση Car

```
public class Car {
    String _company;
    String model;
    int prodyear;
}
```

Ιδιότητες της κλάσης Car

```
public static void main(String[] args) {
    Car x = new Car();
    x._company = "Porsche";
    x.model  = "911";
    x.prodyear = 1975;
    Car m = new Car();
}
```

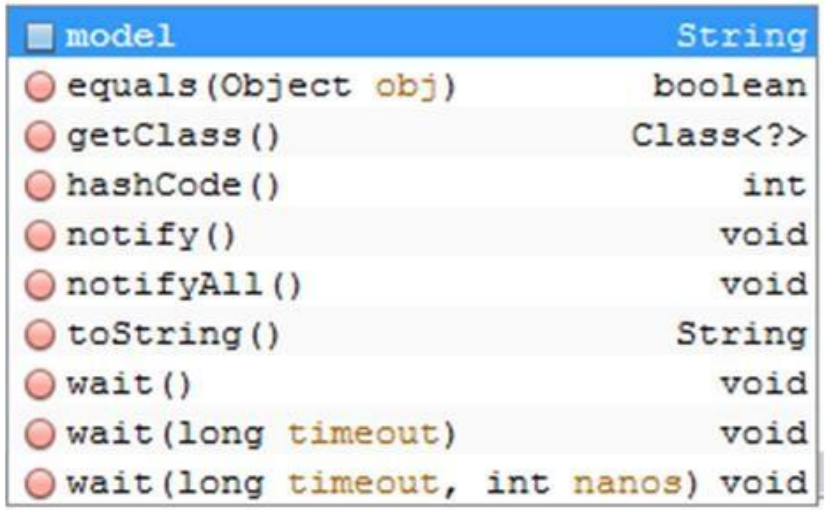


Δημόσια και Ιδιωτική Πρόσβαση

```
public class Car {
   private String company;
   public   String model;
   private int prodyear;
}
```

```
public static void main(String[] args) {
      Car x = new Car();
      x.company = "Porsche";
      x.model  = "911";
      x.prodyear = 1975;
      Car m = new Car();
}
```

```
public static void main(String[] args) {
    Car x = new Car();
    x.
```

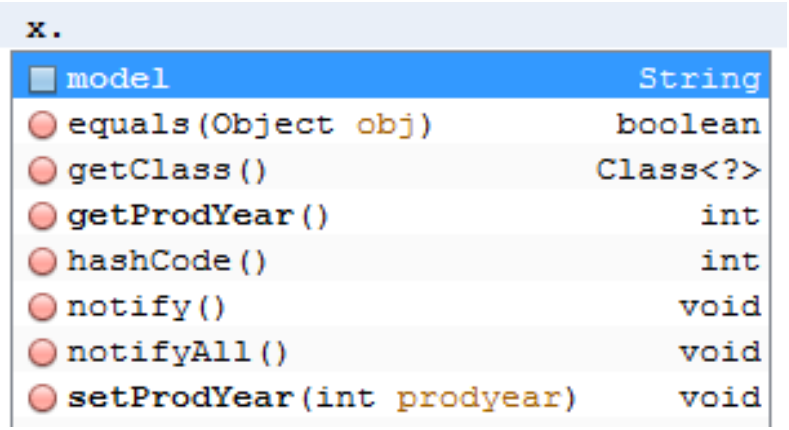


Τα ιδιωτικά (private) μέλη (μέθοδοι και ιδιότητες) δεν είναι ορατά εκτός της κλάσης.

Μέθοδοι

```
public class Car {
  private String company;
  public String model;
  private int prodyear;

  public void setProdYear(int a){
     if(a > 0){
       prodyear = a;
     }
  }

  public int getProdYear(){
     return prodyear;
  }
}
```

```
public class Car {
   private String company;
   public String model;
   private int prodyear;

   public void setProdYear(int prodyear){
      if(prodyear > 0){
        this.prodyear = prodyear;
      }
   }
}
```



Κατασκευαστές (constructors)

```
public class Car {
    private String _company;
    private String model;
    private int prodyear;
}
```

```
public class Car {
    private String _company;
    private String model;
    private int prodyear;

    Car()
    {
    }
}
```

Προκαθορισμένος Κατασκευαστής (default constructor)

```
public static void main(String[] args) {
        Car x = new Car();
}
```

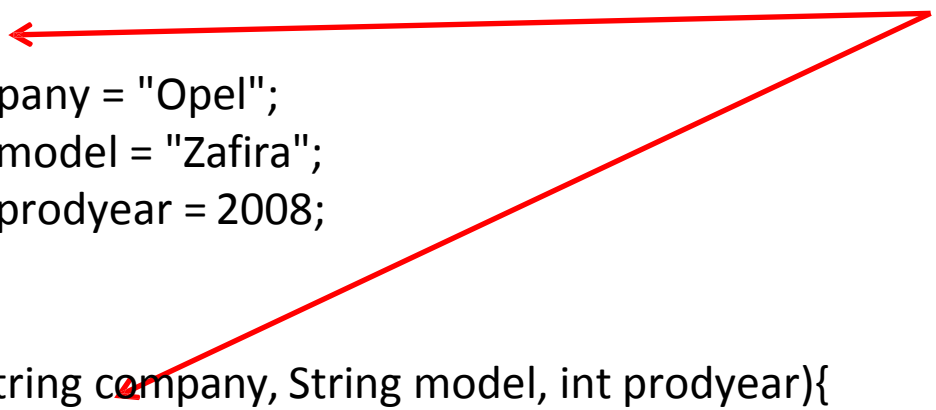
Αφού οι ιδιότητες δεν είναι δημόσιες τότε ο μόνος τρόπος για να τους δώσουμε τιμές είναι μέσω δημοσίων μεθόδων.

```
Car(){
    company = "Opel";
    this.model = "Zafira";
    this.prodyear = 2008;
}

Car(String company, String model, int prodyear){
    company = company;
    this.model = model;
    this.prodyear = prodyear;
}
```

Υπερφόρτωση μεθόδων

```
public class Car {
   private String _company;
   private String model;
   private int prodyear;

   Car(){
     company = "Opel";
     this.model = "Zafira";
     this.prodyear = 2008;
   }

   Car(String company, String model, int prodyear){
     company = company;
     this.model = model;
     this.prodyear = prodyear;
   }
```

Υπερφόρτωση κατασκευαστή



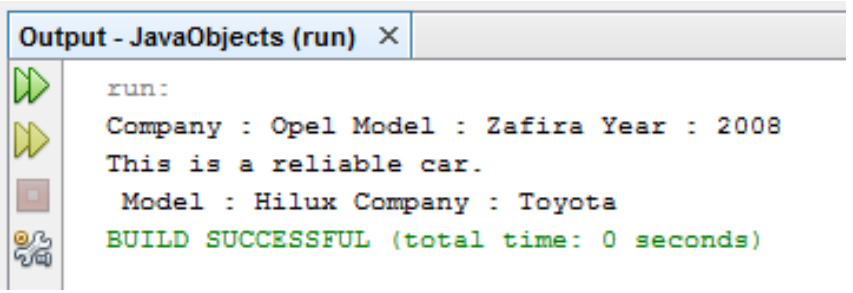
Υπερφόρτωση μεθόδων (συνέχεια)

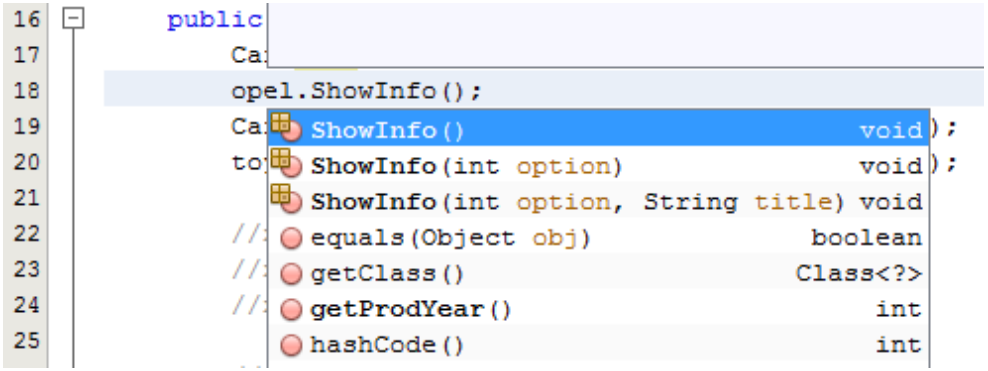
```
void ShowInfo()  {
 System.out.println("Company : " + company + " Model : " + model + " Year : " + prodyear);
 }

 void ShowInfo(int option) {
    if(option == 1)  {
       System.out.println(" Model : " + model + " Company : " + _company);
    }
   else {
       System.out.println("Company : " + company + " Model : " + model);
   }
  }

  void ShowInfo(int option, String title) {
      System.out.println(title);
      ShowInfo(option);
   }
} // end of Car
```

Υπερφόρτωση της μεθόδου **ShowInfo**

Κλήση μεθόδων

```
public class JavaObjects {

    public static void main(String[] args) {
        Car opel = new Car();
        opel.ShowInfo();
        Car toyota = new Car("Toyota", "Hilux", 2002);
        toyota.ShowInfo(1, "This is a reliable car.");
    }
}
```

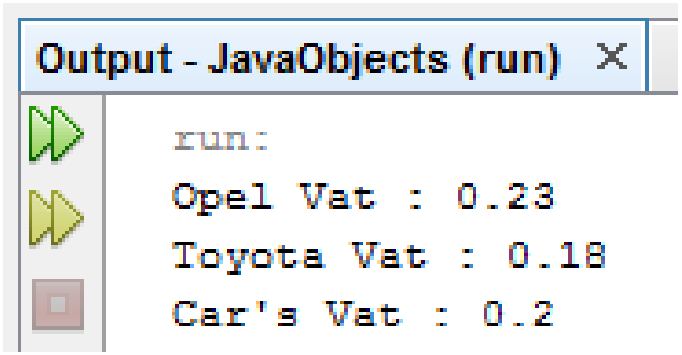




Στατικές ιδιότητες και μέθοδοι

```
public class Car {
    private String _company;
    private String model;
    private int prodyear;
    static float vat = 0.23f;

    // Το υπόλοιπο σώμα της
    // κλάσης παραμένει όπως
    // στις προηγούμενες
    // διαφάνειες
}
```

```
public class JavaObjects {
   public static void main(String[] args) {
        Car opel = new Car();
        System.out.println("Opel Vat : " + opel.vat);

        opel.vat = 0.18f;

        Car toyota = new Car("Toyota", "Hilux", 2002);
        System.out.println("Toyota Vat : " + toyota.vat);
        toyota.vat = 0.20f;

         System.out.println("Car's Vat : " + Car.vat );
    }
}
```



Σε μια στατική ιδιότητα έχουμε πρόσβαση μέσω ενός αντικειμένου της κλάσης αλλά και χωρίς να υπάρχει κάποιο αντικείμενο, κατευθείαν με το όνομα της κλάσης.

ΌνομαΚλάσης.ΟνομαΣτατικήςΙδιότητας

Π.χ. Car.vat = 0.17f;

Στατικές ιδιότητες και μέθοδοι (συν.)

Η δήλωση της **main** σαν **static** είναι απαραίτητη σε μια κλάση για να μπορεί να την εκτελέσει , η μηχανή της Java. Για παράδειγμα, στη παρακάτω κλάση :

```
public class JavaObjects {

    public static void main(String[] args) {
        // σώμα της main
    }
}
```

Πρώτα κάνουμε μεταγλώττιση

javac JavaObjects.java

και μετά κάνουμε εκτέλεση

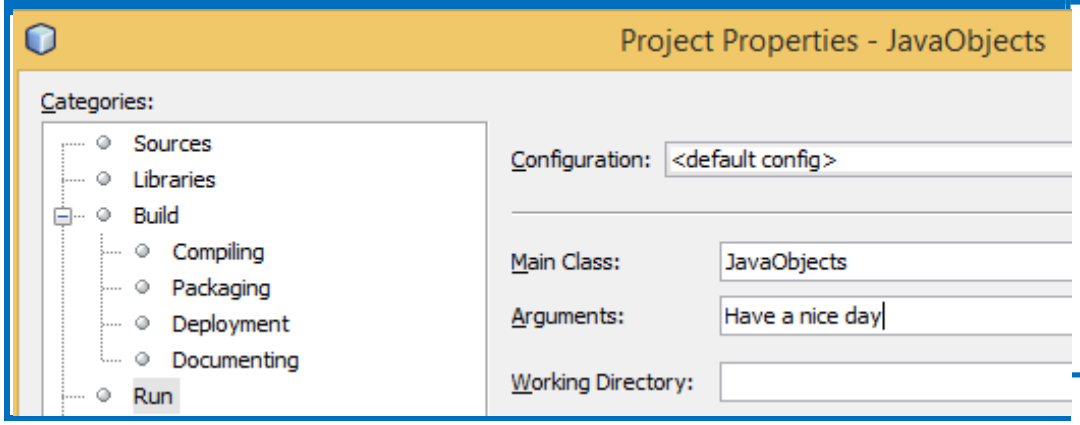
java JavaObjects

Η μηχανή της Java το μόνο που έχει να κάνει, είναι να καλέσει την JavaObjects.main δηλαδή τη μέθοδο main της κλάσης, και να της περάσει σε πίνακα τα ορίσματα της εφαρμογής.

Κλήσεις στο σώμα των στατικών μεθόδων

```
public class JavaObjects {

    private void PrintArguments(String[] args) {
        for(int i = 0; i < args.length; i++) {
            System.out.println(i + ") " + args[i]);
        }
    }

    public static void main(String[] args) {
        PrintArguments(args);
    }
}
```

Error
non-static method **PrintArguments (String[])** cannot be referenced from a static context



Στο NetBeans, δίνουμε παραμέτρους κατά την εκτέλεση του προγράμματος, πατώντας δεξί πλήκτρο στο έργο (όχι στη κλάση) και επιλέγουμε ιδιότητες και στη κατηγορία Run δίνουμε τιμές στο Arguments.

```
C:\test>c:\jdk7\bin\java.exe JavaObjects Have a nice day
0) Have
1) a
2) nice
3) day
```

Στη κονσόλα πληκτρολογούμε τα ορίσματα μετά το όνομα της κλάσης.

Κλήσεις στο σώμα των στατικών μεθόδων (συν.)

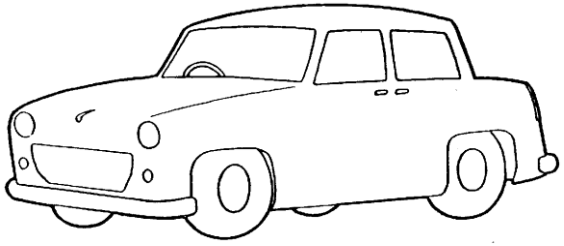
```
public class JavaObjects {
   private static void PrintArguments(String[] args) {
      for(int i = 0; i < args.length; i++) {
         System.out.println(i + ") " + args[i]);
      }
   }

   public static void main(String[] args) {
      PrintArguments(args);
   }
}
```

Κλήση άλλων στατικών μεθόδων

```
public class JavaObjects {
   private void PrintArguments(String[] args) {
      for(int i = 0; i < args.length; i++) {
         System.out.println(i + ") " + args[i]);
      }
   }

   public static void main(String[] args) {
      JavaObjects x = new JavaObjects();
      x.PrintArguments(args);
   }
}
```

Είτε δημιουργία αντικειμένου της κλάσης, μέσα στο σώμα της στατικής μεθόδου, και εν συνεχεία κλήση της μεθόδου του αντικειμένου.

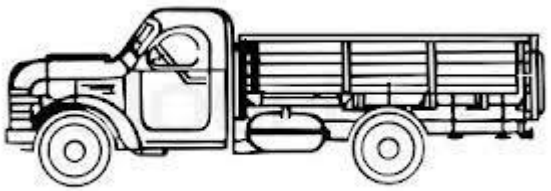
Κληρονομικότητα



Car

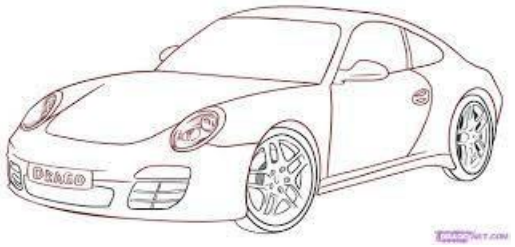
```
public class Car {
   String _company;
   String model;
   int prodyear;
}
```

CargoCar



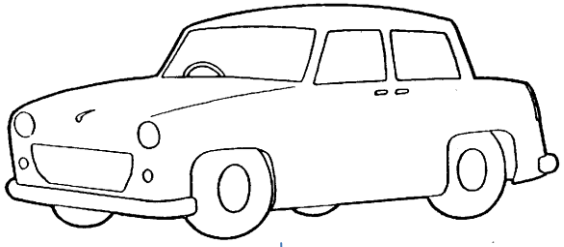
```
public class CargoCar {
   String _company;
   String model;
   int prodyear;

    // Επιπλέον ιδιότητες
    // και μεθόδους
}
```

SportCar



```
public class SportCar {
   String _company;
   String model;
   int prodyear;

    // Επιπλέον ιδιότητες
    // και μεθόδους
}
```

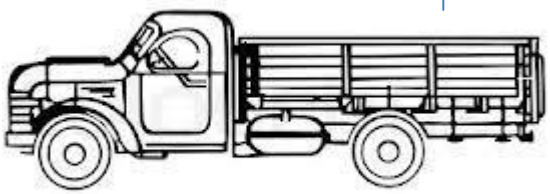
Κληρονομικότητα (συνέχεια)



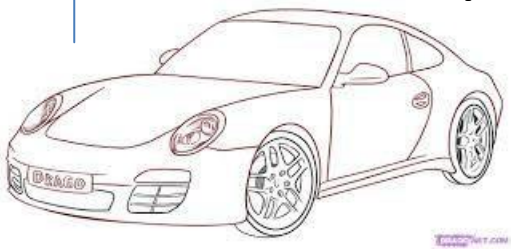
Car

```
public class Car {
    String _company;
    String model;
    int prodyear;
}
```

CargoCar



SportCar



```
public class CargoCar extends Car {

    // Επιπλέον ιδιότητες
    // και μεθόδους
}
```

```
public class SportCar extends Car {

    // Επιπλέον ιδιότητες
    // και μεθόδους
}
```

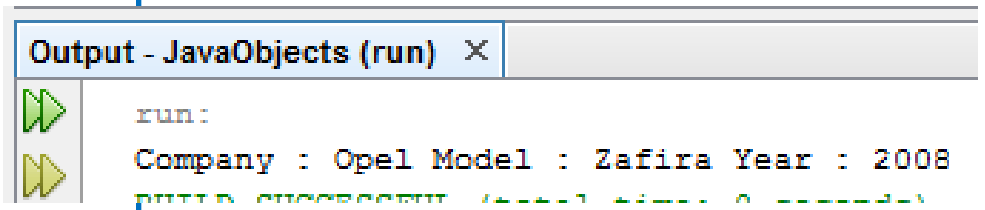
Η κλάση CargoCar

```
public class Car {
   String _company;
   String model;
   int prodyear;

 Car()  {
     _company = "Opel";
     this.model = "Zafira";
     this.prodyear = 2008;
   }

Car(String company, String model, int prodyear)  {
     _company = company;
     this.model = model;
     this.prodyear = prodyear;
   }

void ShowInfo()  {
System.out.println("Company : " + _company +
" Model : " + model + " Year : " + prodyear);
   }
}
```

Η κλάση Car έχει τη συγκεκριμένη μορφή και αγνοούμε τις προηγούμενες διαφάνειες.

```
public class CargoCar extends Car {

}
```

```
public class JavaObjects {

   public static void main(String[] args) {
      SportCar ferrari = new SportCar();
      ferrari.ShowInfo();
    }
}
```



Η κλάση CargoCar (συνέχεια)

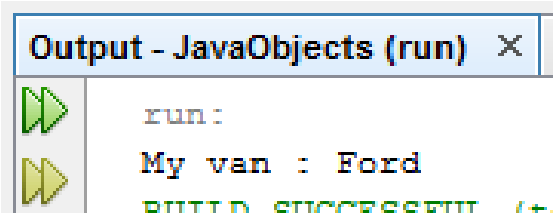
```
public class CargoCar extends Car {
 CargoCar()
   {
     this._company = "Ford";
     this.model = "Transit";
     this.prodyear = 2004;
   }
}
```

Η κλάση Car παραμένει όπως και προηγουμένως

```
public class JavaObjects {

   public static void main(String[] args) {
     CargoCar van = new CargoCar();
     van.ShowInfo();
    }
}
```

Εάν προσθέσουμε στη **CargoCar** τη μέθοδο **ShowInfo** τότε :

```
void ShowInfo()    {
     System.out.println("My van : " + _company);
   }
```



Η κλάση CargoCar (συνέχεια)

```
public class CargoCar extends Car {
   float cargo;

   CargoCar(String company, String model, int prodyear, float cargo)
   {
     _company = company;
     this.model = model;
     this.prodyear = prodyear;
     this.cargo = cargo;
   }
}
```

Προσθέτουμε στη CargoCar την ιδιότητα **cargo** για να αποθηκεύουμε το ωφέλιμο φορτίο που μπορεί να μεταφέρει το φορτηγάκι. Αν δημιουργήσουμε έναν επιπρόσθετο κατασκευαστή, όπως έχουμε και στη Car για να αποθηκεύσουμε τα στοιχεία από το φορτηγάκι, τότε μπορούμε στο σώμα του κατασκευαστή να αποθηκεύσουμε μια προς μια τις τιμές όπως ανωτέρω ή να κάνουμε κλήση το κατασκευαστή της Car.

```
CargoCar(String company, String model, int prodyear, float cargo) {
      super(company, model, prodyear);
      this.cargo = cargo;
   }
```

Abstract Class

```
public abstract class Human {
   String name;
   int year, month, day;

   Human(String name, int year, int month, int day)  {
     this.name = name:
     this.year = year;
     this.month = month;
     this.day = day;
   }

   public void ShowInfo() {
      System.out.println("My name is " + name);
   }
}
```

Δεν επιτρέπεται να δημιουργήσουμε αντικείμενα της κλάσης Human

```
/**
 * @param args the comma
 */
public static void main(
    Human test = new Human("nick", 1955, 2, 10);
```

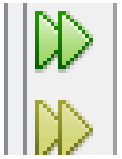
Human is abstract; cannot be instantiated

(Alt-Enter shows hints)

```
public class Student extends Human {

   Student(String name, int year, int month, int day) {
     super(name, year, month, day);
   }
}
```

Μια abstract θα πρέπεινα κληρονομηθεί

```
Student test = new
Student("nick", 1955, 2, 10);
test.ShowInfo();
```



```
run:
My name is nick
```

Abstract μέθοδος

```
public abstract class Human {
   String name;
   int year, month, day;

   Human(String name, int year, int month, int day)  {
      this.name = name;
      this.year = year;
      this.month = month;
      this.day = day;
   }

   public abstract void ShowInfo();
}
```

Από τη στιγμή που μια μέθοδος είναι abstract τότε θα πρέπει και ολόκληρη η κλάση να δηλωθεί σαν abstract.

Οι abstract μέθοδοι δεν έχουν σώμα.

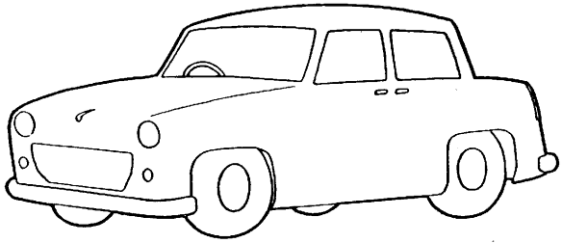
```
public class Student extends Human {

   Student(String name, int year, int month, int day) {
      super(name, year, month, day);
   }
   public void ShowInfo() {
      System.out.println("My name is " + name);
   }
}
```

Η υλοποίηση θα γίνει στις παράγωγες κλάσεις.

```
Student test = new
Student("nick", 1955, 2, 10);
test.ShowInfo();
```

Interfaces

Υλοποίηση με ηλεκτρονικά



Υλοποίηση με κώδικα

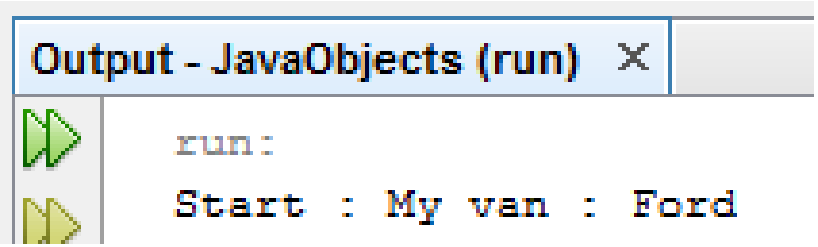
```
public interface Vehicle {
   public void IncreaseSpeed(float incr);
   public void DecreaseSpeed(float decr);
   public void Stop();
   public void Start();
}
```

Interfaces

```
public interface Vehicle {
   public void IncreaseSpeed(float incr);
   public void DecreaseSpeed(float decr);
   public void Stop();
   public void Start();
}
```

```
public class JavaObjects {

    public static void main(String[] args) {
        CargoCar van = new CargoCar();
        van.Start();
    }
}
```

```
public class Car implements Vehicle {


   private float speed = 0;
   public void Start() {
      speed = 10;
      System.out.print("Start :
      "); ShowInfo();
   }
  // Το υπόλοιπο σώμα της Car
  // παραμένει το ίδιο
}
```



Είδος πρόσβασης

Στη Java έχουμε τη δυνατότητα να τροποποιήσουμε τη πρόσβαση στις κλάσεις, στις ιδιότητες, στις μεθόδους και τους κατασκευαστές, γράφοντας μπροστά από το όνομα τους, ένα από τα παρακάτω :

Α) Αν δεν γράψουμε τίποτα τότε είναι ορατά μέσα στο πακέτο.
Β) private – είναι ορατά μόνο μέσα στη κλάση
Γ) public – είναι ορατά από παντού
Δ) protected – είναι ορατά στο πακέτο (package) και σε όλες τις κλάσεις που την κληρονομούν.