

```
package com.example.carpicker;

import android.content.res.AssetManager;
import java.io.InputStream;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class CarBrandList {
    // Δημιουργούμε την λίστα με τις μάρκες
    ArrayList<CarBrand> cbList = new ArrayList<CarBrand>();

    public CarBrandList(AssetManager assets) {
        try {
            //Ο κώδικας διαβάζει ένα XML αρχείο με όνομα "records.xml" από τον φάκελο assets
            // και κάνει parse το περιεχόμενό του για να εξάγει τις πληροφορίες του car brand

            // Εισάγουμε το αρχείο records.xml από τον φάκελο assets, δημιουργούμε ένα
αντικείμενο InputStream
            InputStream is = assets.open("records.xml");

            // Δημιουργούμε ένα νέο DocumentBuilderFactory και DocumentBuilder
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

            // Φορτώνουμε το αρχείο XML στο αντικείμενο dBuilder χρησιμοποιώντας τη μέθοδο
parse(),
            // περνώντας το InputStream με όνομα is.
            Document doc = dBuilder.parse(is);

            // Αποθηκεύει τα αρχικά στοιχεία του αρχείου XML με το όνομα ετικέτας
"carBrand" (περιλαμβάνει την μάρκα και τα μοντέλα)
            // σε ένα NodeList που ονομάζεται nList.
            NodeList nList = doc.getElementsByTagName("carBrand");

            // Επαναλαμβάνεται πάνω από κάθε στοιχείο στο NodeList nList.
            for (int i=0; i< nList.getLength(); i++) {
                // Μέσα στον βρόχο, ανακτά το τρέχον στοιχείο ως αντικείμενο Node με το όνομα
node
                Node node = nList.item(i);

                // Ελέγχει αν ο κόμβος είναι τύπου Element.
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    // Εάν ο κόμβος είναι Element, προχωρά στην εξαγωγή της επωνυμίας και των
μοντέλων της μάρκας αυτοκινήτου.

                    // Ανακτά τους θυγατρικούς κόμβους του στοιχείου "όνομα" και τους
αποθηκεύει σε ένα NodeList με το όνομα nameNode.
                    NodeList nameNode = ((Element) node).
                        getElementsByTagName("name").item(0).getChildNodes();

                    // Εξάγει το όνομα της επωνυμίας από τον πρώτο θυγατρικό κόμβο του
nameNode και το αποθηκεύει σε μια μεταβλητή String που ονομάζεται brand.
                    String brand = nameNode.item(0).getNodeValue();
```

```
// Ανακτά τους θυγατρικούς κόμβους του στοιχείου "models" και τους
αποθηκεύει σε μια NodeList που ονομάζεται modelsNode
NodeList modelsNode = ((Element) node).
    getElementsByTagName("models").item(0).getChildNodes();
// Εξάγει τα μοντέλα της μάρκας αυτοκινήτου από τον πρώτο θυγατρικό κόμβο
του modelsNode και τα αποθηκεύει σε μια μεταβλητή String με το όνομα models.
String models = modelsNode.item(0).getNodeValue();

// Δημιουργεί ένα νέο αντικείμενο CarBrand χρησιμοποιώντας την επωνυμία
και τις τιμές μοντέλων και το προσθέτει σε μια λίστα με το όνομα cbList
cbList.add(new CarBrand(brand, models));
}

// Ο βρόχος συνεχίζεται μέχρι να υποβληθούν σε επεξεργασία όλα τα στοιχεία
στο nList NodeList.
}

} catch (Exception e) {
    e.printStackTrace();
}

}

public void addCarBrand(CarBrand b) {
    cbList.add(b);
}

/*
    public void addModel(String b, String m) {
        for (int i=0; i<cbList.size(); i++) {
            if (cbList.get(i).hasName(b)) {
                cbList.get(i).addModel(m);
            }
        }
    }
*/

// η μέθοδος getAllModelsAsString αναζητά μια συγκεκριμένη μάρκα αυτοκινήτου στη λίστα
cbList
// και ανακτά τα μοντέλα που σχετίζονται με αυτήν τη μάρκα ως συμβολοσειρά.
public String getAllModelsAsString(String b) {
    String s = "";
    for (int i=0; i<cbList.size(); i++) {
        // Ελέγχει εάν το τρέχον αντικείμενο CarBrand (cbList.get(i)) έχει το ίδιο όνομα
        // με την παράμετρο εισόδου b καλώντας τη μέθοδο hasName της κλάσης CarBrand.
        if (cbList.get(i).hasName(b)) {
            // Εάν το τρέχον αντικείμενο CarBrand έχει το καθορισμένο όνομα,
            // ενημερώνει τη μεταβλητή s καλώντας τη μέθοδο getAllModelsAsString
            // της κλάσης CarBrand για να ανακτήσει τα μοντέλα που σχετίζονται
            // με αυτήν την επωνυμία ως συμβολοσειρά.
            s = cbList.get(i).getAllModelsAsString();
        }
    }
    return s;
}

// η μέθοδος getAllBrands εξάγει τα ονόματα όλων των επωνυμιών αυτοκινήτων
```

```
// από τη λίστα cbList και τα επιστρέφει ως ξεχωριστή λίστα συμβολοσειρών.  
public List<String> getAllBrands() {  
    List<String> temp = new ArrayList<String>();  
    for (int i=0; i<cbList.size(); i++) {  
        // ανακτά το όνομα του τρέχοντος αντικειμένου CarBrand (cbList.get(i))  
        // καλώντας τη μέθοδο getName της κλάσης CarBrand και την προσθέτει στην temp.  
        temp.add(cbList.get(i).getName());  
    }  
    return temp;  
}  
  
// η μέθοδος getAllModels αναζητά μια συγκεκριμένη μάρκα αυτοκινήτου στη λίστα cbList  
// και ανακτά τα μοντέλα που σχετίζονται με αυτήν τη μάρκα ως λίστα συμβολοσειρών  
public List<String> getAllModels(String b) {  
    List<String> temp = new ArrayList<String>();  
    for (int i=0; i<cbList.size(); i++) {  
        // Ελέγχει εάν το τρέχον αντικείμενο CarBrand (cbList.get(i)) έχει το ίδιο όνομα  
        // με την παράμετρο εισόδου b καλώντας τη μέθοδο hasName της κλάσης CarBrand.  
        // Εάν το τρέχον αντικείμενο CarBrand έχει το καθορισμένο όνομα,  
        // ενημερώνει τη μεταβλητή temp καλώντας τη μέθοδο getAllModels της κλάσης  
CarBrand  
        // για να ανακτήσει τα μοντέλα που σχετίζονται με αυτήν την επωνυμία ως λίστα  
συμβολοσειρών.  
        if (cbList.get(i).hasName(b)) {  
            temp = cbList.get(i).getAllModels();  
        }  
    }  
    return temp;  
}  
}
```