Ελληνικό Μεσογειακό Πανεπιστήμιο
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 5

Καπετανάκης Φανούριος

Τι θα δούμε

- Κληρονομικότητα – πολυμορφισμός
- Αφηρημένες κλάσεις-μέθοδοι (abstract)
- Interfaces

Παράδειγμα (1)

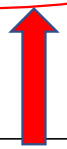
Κληρονομικότητα μεθόδων

```
package inheritance;

public class Vehicle {
    private int cubicCentimetres;

    public Vehicle(int cubicCentimetres) {
    this.cubicCentimetres=cubicCentimetres;
    }



    public int getCubicCentimetres() {
return cubicCentimetres;
}



public void printInfo() {
    System.out.println("Τα κυβικά είναι:");
    }
}
```

```
package inheritance;

public class Car extends Vehicle{

private String brand;
private String gasoline;

//οι κατασκευαστές δεν κληρονομούνται
//αυτό σημαίνει ότι πρέπει να υλοποιηθούν στις υποκλάσεις

//κατασκευαστής υποκλάσης
public Car(int cc, String brand, String gasoline) {
super(cc);//κλήση του κατασκευαστή της υπερκλάσης
this.brand = brand;
this.gasoline = gasoline;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println(getCubicCentimetres()+" Το αυτοκίνητο είναι "+brand
+" και κινείται με "+gasoline);
}

}
```

Παράδειγμα (1)

```
package inheritance;

public class Motorcycle extends Vehicle{

private String brand;
private String electricMachinery;

public Motorcycle(int cc, String brand,String electricMachinery) {
super(cc);
this.brand = brand;
this.electricMachinery = electricMachinery;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println(getCubicCentimetres()+" Η μηχανή είναι "+brand +" και
κινείται με "+electricMachinery);
}

}
```



```
package inheritance;

public class Main {

public static void main(String[] args) {

Car C1 = new Car(1200,"Ford","Βενζίνη");

Motorcycle MC1 =new Motorcycle(150,"Piaggio","Ρεύμα");

            C1.printInfo();
            MC1.printInfo();


}

}
```

Output:

```
Τα κυβικά είναι :
1200 Το αυτοκίνητο είναι Ford και κινείται με Βενζίνη
Τα κυβικά είναι :
150 Η μηχανή είναι Piaggio και κινείται με Ρεύμα
```

Protected

- Όταν μια ιδιότητα σε μια κλάση δηλωθεί σαν private, είναι ιδιωτική για τα αντικείμενα όλων των κλάσεων ακόμα και των απογόνων της.
- Εναλλακτικός τρόπος είναι να θέσουμε το προσδιοριστικό ορατότητας protected σε ένα πεδίο (προστατευμένη ιδιότητα).
- αυτό σημαίνει ότι παραμένει μη ορατό το πεδίο σε αντικείμενα άλλων κλάσεων, είναι όμως ορατό στους απόγονους της κλάσης.

Παράδειγμα (2) - protected

Πρόσβαση υποκλάσεων σε μεταβλητές υπερκλάσης, χωρίς μέθοδο get.

```
package inheritance;

public class Vehicle {
    protected int cubicCentimetres;

    public Vehicle(int cubicCentimetres) {
    this.cubicCentimetres=cubicCentimetres;
    }



    public int getCubicCentimetres() {
return cubicCentimetres;
}



public void printInfo() {
    System.out.println("Τα κυβικά είναι:"
+cubicCentimetres);
    }
}
```

```
package inheritance;

public class Car extends Vehicle{

private String brand;
private String gasoline;

//οι κατασκευαστές δεν κληρονομούνται
//αυτό σημαίνει ότι πρέπει να υλοποιηθούν στις υποκλάσεις

//κατασκευαστής υποκλάσης
public Car(int cc, String brand, String gasoline) {
super(cc);//κλήση του κατασκευαστή της υπερκλάσης
this.brand = brand;
this.gasoline = gasoline;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println(cubicCentimetres+" Το αυτοκίνητο είναι "+brand +" και
κινείται με "+gasoline);
}

}
```

Παράδειγμα (2) - protected

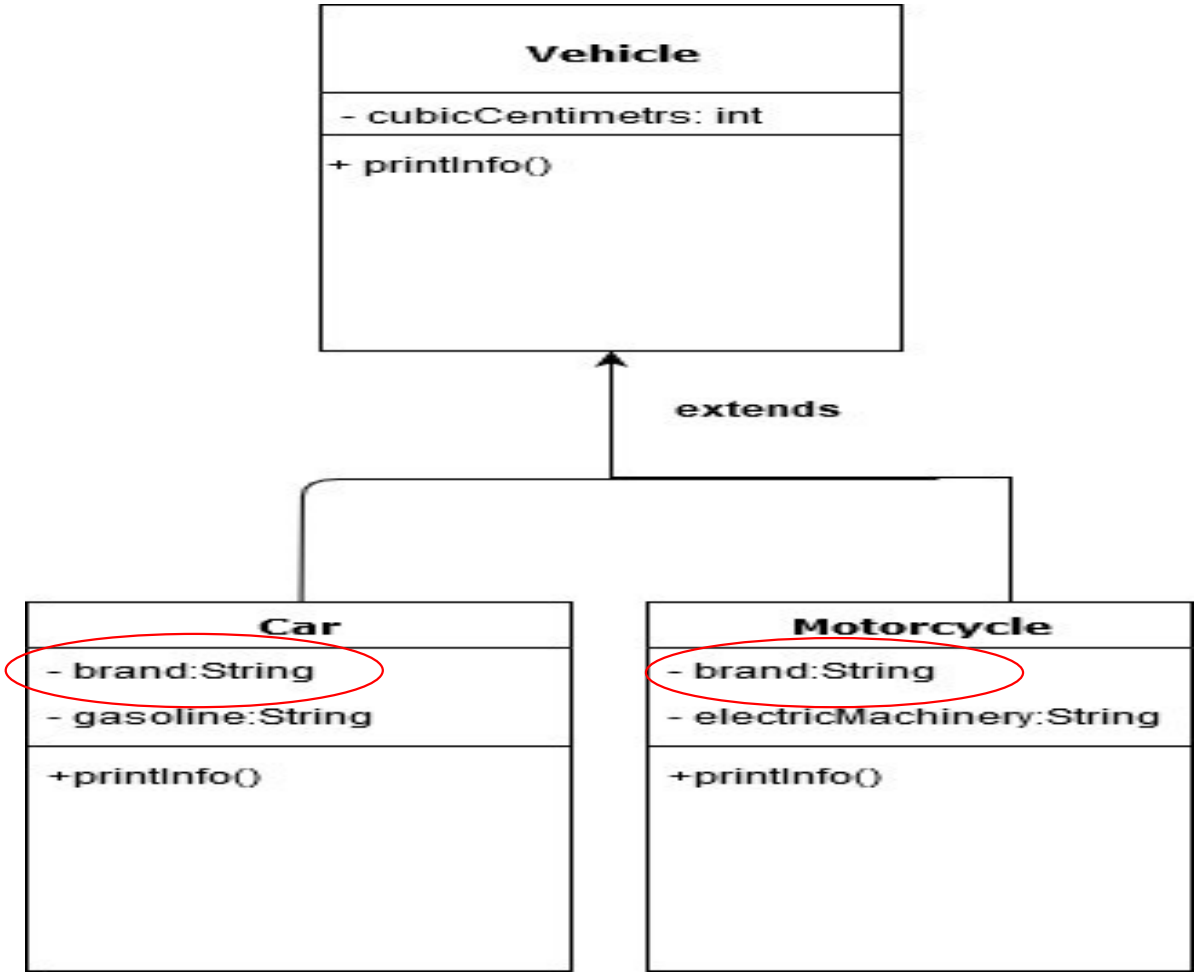
```java
package inheritance;

public class Motorcycle extends Vehicle{

private String brand;
private String electricMachinery;

public Motorcycle(int cc, String brand,String electricMachinery) {
super(cc);
this.brand = brand;
this.electricMachinery = electricMachinery;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println("cubicCentimetres+ Η μηχανή είναι "+brand +" και
κινείται με "+electricMachinery);
}

}
```

```java
package inheritance;

public class Main {

public static void main(String[] args) {

Car C1 = new Car(1200,"Ford","Βενζίνη");

Motorcycle MC1 =new Motorcycle(150,"Piaggio","Ρεύμα");

            C1.printInfo();
            MC1.printInfo();


}

}
```

Output:

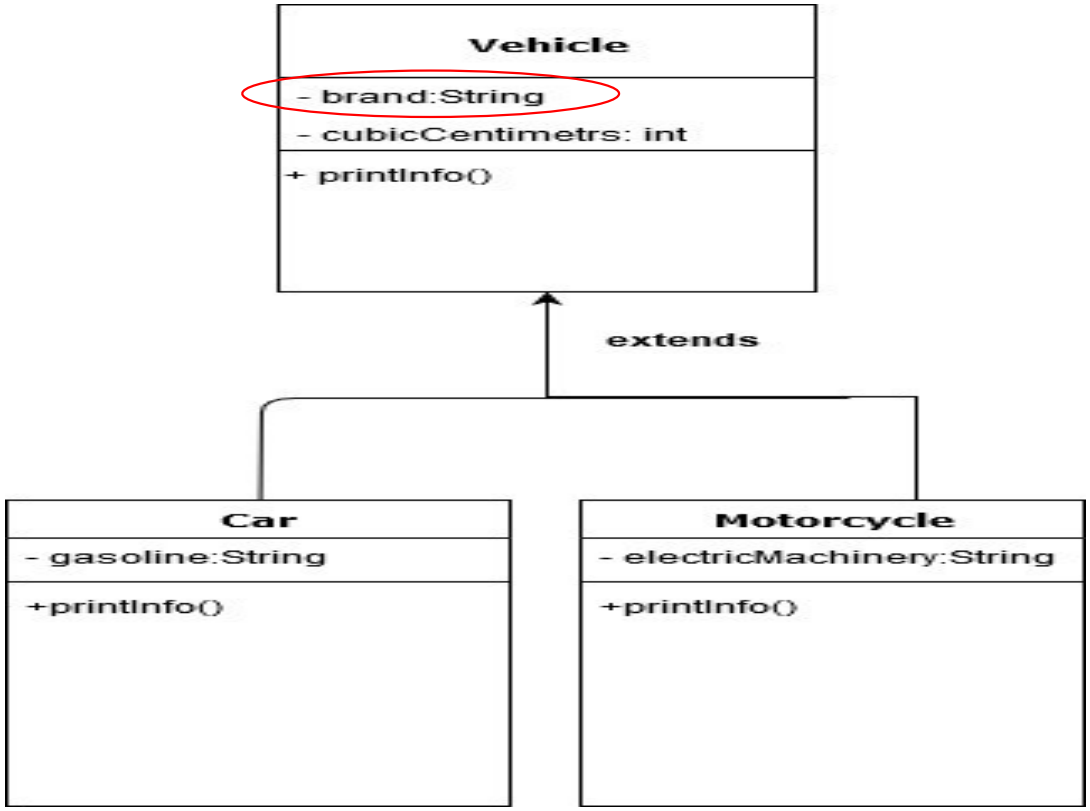
```
Τα κυβικά είναι :
1200 Το αυτοκίνητο είναι Ford και κινείται με Βενζίνη
Τα κυβικά είναι :
150 Η μηχανή είναι Piaggio και κινείται με Ρεύμα
```

Ιεραρχία κλάσεων προηγούμενου παραδείγματος



Λάθος σχεδίαση
Επανάληψη πληροφορίας

Λύση



Τα κοινά χαρακτηριστικά τα ομαδοποιούμε στην υπερκλάση για να κληρονομηθούν από τις υποκλάσεις.

Υλοποίηση της λύσης σε κώδικα

```
package inheritance;

public class Vehicle {
private String brand;
private int cubicCentimetres;

    public Vehicle(String brand,int cubicCentimetres) {
    this.brand = brand;
    this.cubicCentimetres=cubicCentimetres;
    }



    public int getCubicCentimetres() {
return cubicCentimetres;
}



public void printInfo() {
System.out.println("Η μάρκα είναι: "+brand );
        System.out.println("Τα κυβικά είναι:
"+cubicCentimetres );
    }
}
```

```
package inheritance;

public class Car extends Vehicle{


private String gasoline;

//οι κατασκευαστές δεν κληρονομούνται
//αυτό σημαίνει ότι πρέπει να υλοποιηθούν στις υποκλάσεις

//κατασκευαστής υποκλάσης
public Car( String brand,int cc, String gasoline) {
super(brand,cc);//κλήση του κατασκευαστή της υπερκλάσης
this.gasoline = gasoline;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println("και κινείται με: "+gasoline);
}

}
```

Υλοποίηση της λύσης σε κώδικα

```java
package inheritance;

public class Motorcycle extends Vehicle{


private String electricMachinery;

public Motorcycle(String brand, int cc, String electricMachinery) {
super(brand,cc);
this.electricMachinery = electricMachinery;
}

public void printInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println(" και κινείται με: "+electricMachinery);
}

}
```

```java
package inheritance;

public class Main {

public static void main(String[] args) {



Car C1 = new Car("Ford",1200,"Βενζίνη");
C1.printInfo();


Motorcycle MC1 = new  Motorcycle("Piaggio",150,"Ρεύμα");

MC1.printInfo();
}

}
```

Output:

```
Τα κυβικά είναι: 1200
και κινείται με: Βενζίνη
Η μάρκα είναι: Piaggio
Τα κυβικά είναι: 150
και κινείται με: Ρεύμα
```

Η java μας επιτρέπει να δηλώσουμε αντικείμενα χωρίς να τα ορίσουμε
Παράδειγμα: **`Vehicle v;`**

```
package inheritance;

public class Main {

public static void main(String[] args) {

        Vehicle V1;//αναφορά τύπου Vehicle

//Ανάθεση αναφοράς V1 σε αντικείμενο Car
V1 = new Car("Ford",1200,"Βενζίνη");
V1.printInfo();

//Ανάθεση αναφοράς V1 σε αντικείμενο Motorcycle
V1 = new  Motorcycle("Piaggio",150,"Ρεύμα");

V1.printInfo();
}

}
```

Output:

```
Η μάρκα είναι: Ford
Τα κυβικά είναι: 1200
και κινείται με: Βενζίνη
Η μάρκα είναι: Piaggio
Τα κυβικά είναι: 150
και κινείται με: Ρεύμα
```

Πολυμορφισμός

```
public class Main {

public static void main(String[] args) {

        Vehicle V1;//αναφορά τύπου Vehicle

        for(int i =0; i<2; i++) {
        Scanner input = new Scanner(System.in);
        int answer;

System.out.println("Τι είδους όχημα ? (1:αυτοκίνητο, 2:μοτοσυκλέτα)");
answer = input.nextInt();

      if(answer ==1)
          V1 = new Car("Ford",1200,"Βενζίνη");
      else
          V1 = new Motorcycle("Piaggio",150,"Ρεύμα");
       //πολυμορφική κλήση
        V1.printInfo();
        }

  }

}
```

Output:

```
Τι είδους όχημα ? (1:αυτοκίνητο, 2:μοτοσυκλέτα)
1
Η μάρκα είναι: Ford
Τα κυβικά είναι: 1200
και κινείται με: Βενζίνη
Τι είδους όχημα ? (1:αυτοκίνητο, 2:μοτοσυκλέτα)
2
Η μάρκα είναι: Piaggio
Τα κυβικά είναι: 150
και κινείται με: Ρεύμα
```

Αφηρημένες μέθοδοι

Όπως μπορούμε να δηλώσουμε αντικείμενα χωρίς να τα ορίσουμε, κατά τον ίδιο τρόπο μπορούμε δηλώσουμε μια μέθοδο χωρίς να την ορίσουμε:

```
public abstract void speak();
```

Μπορεί να έχει υπογραφή και επιστρεφόμενο τύπο, λείπει όμως το σώμα της μεθόδου. Αυτές οι μέθοδοι ονομάζονται **abstract**

Αφηρημένες κλάσεις

Όταν μια κλάση έχει μια αφηρημένη μέθοδο, πρέπει να είναι και αυτή αφηρημένη κλάση (δηλαδή πρέπει να την δηλώσουμε abstract).

Μια αφηρημένη κλάση δεν μπορεί να έχει αντικείμενα.

Μια αφηρημένη κλάση μπορεί να έχει υλοποιημένες μεθόδους ή/και μη υλοποιημένες μεθόδους.

Μπορούμε να δηλώσουμε μια κλάση abstract και χωρίς να έχει κάποια μέθοδο abstract μόνο και μόνο για να μην μπορεί ο χρήστης να δημιουργήσει αντικείμενα αυτής της κλάσης.

Παράδειγμα abstract κλάσης

Όπως είδαμε στο προηγούμενο εργαστήριο, δηλώσαμε την abstract κλάση Pet με 2 υποκλάσεις την Dog και την Cat

Δεν έχει νόημα να υπάρχουν αντικείμενα της κλάσης Pet γιατί είναι αφηρημένη έννοια (τι είδους Pet είναι;)

Υπάρχει όμως νόημα να δηλώσουμε ένα αντικείμενο της κλάσης Dog που είναι υποκλάση της Pet.

Παράδειγμα (3)

Δημιουργία **abstract** (αφηρημένης) κλάσης **Pet** με **μέθοδο speak**, στην οποία δε μπορούμε να δημιουργήσουμε αντικείμενα με δύο υποκλάσεις, την Dog και Cat.

```
package inheritance2;

public abstract class Pet {
    private String name;

    public Pet(String name) {
    this.name = name;
    }

public String getName() {
return name;
}
  public abstract void speak();

}
```

```
package inheritance2;

public class  Dog extends Pet {

private String age;

public Dog(String name, String age) {
super(name);
this.age = age;
}

//implements
public void speak() {
System.out.println("Woof Woof!");
}

public void printInformation(){
System.out.println("My name is:
"+getName());
System.out.println("age " +age);
}
}
```

```
package inheritance2;

public class Cat extends Pet {

private String age;

public Cat(String name, String age) {
super(name);
this.age = age;
}


//implements
public void speak() {
System.out.println("Miaou Miaou!");
}

public void printInformation(){
System.out.println("My name is:
"+getName());
System.out.println("age " +age);
}
}
```

Παράδειγμα (3)

```
package inheritance2;

public class Main {

public static void main(String[] args) {

Dog D1 = new Dog("Rex","7");
Cat C1 = new Cat("Tom","10");

      D1.speak();
      C1.speak();

      D1.printInformation();
      C1.printInformation();

}

}
```

Output:

```
Woof Woof!
Miaou Miaou!
My name is: Rex
age 7
My name is: Tom
age 10
```

Interfaces – διασυνδέσεις

Σε ένα interface δηλώνουμε τις μεθόδους χωρίς να περιέχουν κάποια υλοποίηση (χωρίς σώμα).

Μια κλάση interface δεν μπορεί να δημιουργήσει αντικείμενα.

Χρειάζεται να υλοποιηθεί (implements) από κάποια άλλη κλάση.

Αυτό σημαίνει ότι πρέπει να φροντίσουμε μέσα σε κάποια κλάση να υλοποιήσουμε όλες τις μεθόδους που δηλώσαμε στο interface.

Μπορεί να έχει σταθερές και final μεταβλητές.

Μερική υλοποίηση interface

Μπορούμε να ορίσουμε κάποιες από τις μεθόδους σε ένα interface, άλλα όχι όλες.

Αν μια κλάση δεν υλοποιεί όλες τις μεθόδους, είναι σαν **αφηρημένη** κλάση και πρέπει να την δηλώσουμε **abstract.**

Μπορούμε να επεκτείνουμε (extends) ένα interface για να του προσθέσουμε μεθόδους.

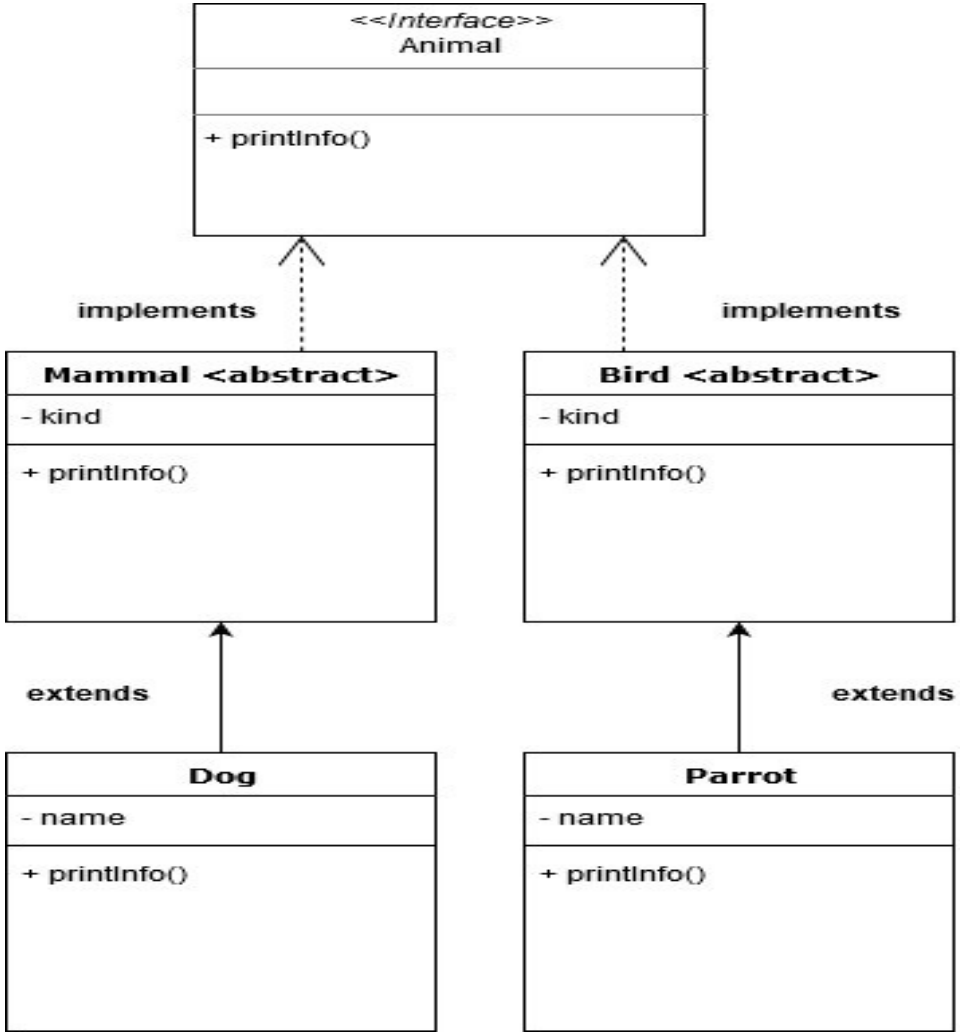
Γιατί interfaces;

Μια κλάση μπορεί να κληρονομήσει μόνο μια άλλη κλάση αλλά μπορεί να υλοποιεί πολλά interfaces (πολλαπλή κληρονομικότητα).

Επίσης μπορούμε να γράφουμε μεθόδους που χρησιμοποιούνται σε περισσότερες από μία κλάσεις.

Χρειάζονται στα γραφικά.

Παράδειγμα (4) - interface



- Ένας σκύλος και ένας παπαγάλος είναι και τα δύο ζώα, αλλά ανήκουν σε διαφορετικά είδη ζώων.
- Αν θέλουμε να φτιάξουμε αντικείμενα τύπου Dog ή τύπου Parrot, το διπλανό σχήμα είναι μία πιθανή λύση.

Παράδειγμα (4) - interface

```java
public interface Animal {

public void printInfo();

}
```

```java
public abstract class Mammal implements Animal {

private String kind;

//Constructor
public Mammal(String kind){
this.kind=kind;
}

public void printInfo(){
    System.out.println("Είμαι "+ kind);
  }
}
```

Παράδειγμα (4) - interface

```
public class Dog extends Mammal {

private String name;

public Dog(String akind, String name){
super(akind);
this.name=name;
}


public void printInfo(){
super.printInfo();
System.out.println("και ονομάζομαι "+name);
}
}
```

```
public abstract class Bird implements Animal{

private String kind;

public Bird(String kind){
this.kind = kind;
}
     public void printInfo(){
     System.out.println("Είμαι "+kind);
     }

}
```

Παράδειγμα (4) - interface

```
public class Parrot extends Bird {

private String name;

 public Parrot(String aKind,String name){
  super(aKind);
  this.name=name;
  }

  public void printInfo(){
  super.printInfo();
  System.out.println("και ονομάζομαι "+name);
  }
}
```

```
public class Main {

public static void main(String[] args) {
Dog dog = new Dog("Θηλαστικό","Rex");
Parrot parrot = new Parrot("Πουλί","Donald");
        dog.printInfo();
        parrot.printInfo();

}

}
```

Output:

```
Είμαι Θηλαστικό
και ονομάζομαι Rex
Είμαι Πουλί
και ονομάζομαι Donald
```

Άσκηση για παρουσίες μαθήματος

Να δημιουργήσετε μια κλάση interface την Food με μια μέθοδο την kindFood και δύο κλάσεις, την Fruit και την Vegetable, που θα κάνουν implements την Food. Η κλάση Fruit έχει δύο πεδία τύπου String, το color και το name, και η Vegetable έχει ένα πεδίο τύπου String, το name, και ένα πεδίο τύπου double, το calories. Η μέθοδος kindFood στις κλάσεις Fruit και Vegetable να τροποποιηθεί έτσι ώστε να εμφανίζει στην οθόνη τα πεδία της κάθε κλάσης. Στην main να δημιουργήσετε ένα αντικείμενο τύπου Fruit και ένα αντικείμενο τύπου Vegetable και να εμφανίζονται τα πεδία τους.