Ελληνικό Μεσογειακό Πανεπιστήμιο
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 4

Καπετανάκης Φανούριος

Πέρασμα παραμέτρων σε μεθόδους

```
class Car
{
    private int position = 0;

    public void moveManySteps(int steps, String direction)
    {
       if (direction.equals("right"){ position += steps;}
       if (direction.equals("left") { position -= steps;}
    }
}

class MovingCar3
{
    public static void main(String args[]){
       Car myCar = new Car();
       myCar.moveManySteps(10, "left");
    }
}
```

Μέθοδος με πολλές παραμέτρους

Τα ορίσματα θα πρέπει να συμφωνούν με το πλήθος και τους τύπους των παραμέτρων στην αντίστοιχη θέση

Κλήση της μεθόδου

Παράδειγμα (1)

Στο παρακάτω παράδειγμα δημιουργούμε μία κλάση Car που έχει μια μέθοδο, την findCar, η οποία παίρνει ως ορίσματα τρία αντικείμενα τύπου Car, αναζητάει για συγκεκριμένο αυτοκίνητο, με βάση τη μάρκα και τα κυβικά, και τυπώνει κατάλληλο μήνυμα.

```
package lab4;

import java.util.Scanner;

public class Car {
private String brand;
private String cubicCentimetres;

//Constructor
public Car(String brand, String cubicCentimetres ) {
this.brand = brand;
this.cubicCentimetres = cubicCentimetres;
}

public String getBrand() {
return brand;
}

public String getCubicCentimetres() {
return cubicCentimetres;
}

public void findCar(Car x, Car y, Car z ) {
Scanner input = new Scanner(System.in);
System.out.println("δώσε όνομα αυτοκινήτου που ψάχνεις: ");

String theBrand = input.nextLine();

System.out.println("δώσε κυβικά που ψάχνεις: ");

String theEngineCapacity = input.nextLine();

if((theBrand.equals(x.brand))&&(theEngineCapacity.equals(x.cubicCent
imetres))) {
System.out.println("το αυτοκίνητο που ψάχνεις είναι:  "+x.brand
+" με "+x.cubicCentimetres+" κυβικά");
}
else
if((theBrand.equals(y.brand))&&(theEngineCapacity.equals(y.cubicCent
imetres))) {
System.out.println("το αυτοκίνητο που ψάχνεις είναι:  "+y.brand
+" με "+y.cubicCentimetres+" κυβικά");

  }

else
if((theBrand.equals(z.brand))&&(theEngineCapacity.equals(z.cubicCent
imetres))) {
System.out.println("το αυτοκίνητο που ψάχνεις είναι: "+z.brand
+" με "+z.cubicCentimetres+" κυβικά");

    }

    else
    System.out.println("Το αυτοκίνητο που ψάχνεις δεν βρέθηκε");
}
}
```

Παράδειγμα (1)

```
package lab4;

public class Main {

public static void main(String[] args) {

Car C1 = new Car("Toyota","1200");
Car C2 = new Car("Ford","1400");
Car C3 = new Car("Fiat","1000");

C1.findCar(C1, C2, C3);
}

}
```

Output:

```
δώσε όνομα αυτοκινήτου που ψάχνεις:
Ford
δώσε κυβικά που ψάχνεις:
1400
το αυτοκίνητο που ψάχνεις είναι:  Ford με 1400 κυβικά
```

Στατικές μέθοδοι

- Για να καλέσουμε μια μέθοδο μιας κλάσης θα πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης
- Εξαίρεση: οι στατικές μέθοδοι
  - Μπορούν να κληθούν χρησιμοποιώντας το όνομα της κλάσης
- Παράδειγμα:
  - Η μέθοδος main που έχουμε δει καλείται χωρίς να έχουμε δημιουργήσει αντικείμενο
  - Γι αυτό και πρέπει να οριστεί ως static.

Παράδειγμα (2)

Μπορούμε να φτιάξουμε μία συνάρτηση, η οποία να κάνει ένα μαθηματικό υπολογισμό, χωρίς να χρειάζεται να δημιουργήσουμε ένα αντικείμενο για να την καλέσουμε.

```
public class MethodOverload {

public static void main(String[] args) {

System.out.println("Το τεράγωνο του 7.5 είναι: "+square(7.5));
System.out.println("Το τετράγωνο του 7 είναι: "+square(7));

}
    //η μέθοδος square με ακέραιο όρισμα.
public static int square(int intValue){
System.out.println("Έγινε κλήση με ακέραιο όρισμα: "+intValue);

return intValue * intValue;
}

//η μέθοδος square με δεκαδικό όρισμα.
//Overloading
    public static double square(double doubleValue){
    System.out.println("Έγινε κλήση με δεκαδικό όρισμα: "+doubleValue);

    return doubleValue * doubleValue;
    }
}
```

Overloading: *χρήση ίδιας μεθόδου με διαφορετικό πλήθος/ είδος ορισμάτων*

Output:

```
Έγινε κλήση με δεκαδικό όρισμα: 7.5
Το τεράγωνο του 7.5 είναι: 56.25
Έγινε κλήση με ακέραιο όρισμα: 7
Το τετράγωνο του 7 είναι: 49
```

ArrayList

- Constructors
  - `ArrayList<T> myList = new ArrayList<T>();`
- Μέθοδοι
  - `add(T x)`: προσθέτει το στοιχειό `x` στο τέλος του πίνακα.
  - `add(int i, T x)`: προσθέτει το στοιχειό `x` στη θέση `i` και μετατοπίζει τα υπόλοιπα στοιχεία κατά μια θέση.
  - `get(int i)`: επιστρέφει το αντικείμενο τύπου `T` στη θέση `i`.
  - `remove(int i)`: αφαιρεί το στοιχείο στη θέση `i`
  - `remove(T x)`: αφαιρεί το στοιχείο `x`
  - `set(int i, T x)`: θέτει στην θέση `i` την τιμή `x` αλλάζοντας την προηγούμενη
  - `size()`: ο αριθμός των στοιχείων του πίνακα.
- Διατρέχοντας τον πίνακα:
  - `ArrayList<T> myList = new ArrayList<T>();`
  - `for(T x: myList){…}`

Παράδειγμα (3)

ArrayList: Είναι ένας δυναμικός πίνακας και ως προς το μέγεθος και ως προς το είδος. Μπορούμε να δημιουργήσουμε ένα ArrayList με αντικείμενα κλάσης δικού μας τύπου.

```
package exampleArrayList;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {
ArrayList<String> cars = new ArrayList<String>();

    cars.add("Volvo");//προσθήκη στοιχείου στο cars
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    for (int i = 0; i < cars.size(); i++) {
        System.out.println(cars.get(i));
      }

    cars.remove(2);//Σβήνει το 3ο στοιχείο του ArrayList

    System.out.println("μετά την διαγραφή του στοιχείου, το ArrayList:");

    //Βελτιωμένη πρόταση for
    for (String car : cars) {
        System.out.println(car);
      }

}

}
```

Output:

```
Volvo
BMW
Ford
Mazda
μετά την διαγραφή του στοιχείου, το ArrayList:
Volvo
BMW
Mazda
```

Βελτιωμένη πρόταση for

- η βελτιωμένη πρόταση for διασχίζει τα στοιχεία ενός πίνακα, χωρίς τη χρήση ενός μετρητή αποφεύγοντας έτσι την πιθανότητα να βγούμε έξω απ' τον πίνακα και να πάρουμε Null Pointer Exception
- for (παράμετρος: όνομα πίνακα)

  {πρόταση}
- η βελτιωμένη πρόταση for μπορεί να χρησιμοποιηθεί μόνο για λήψη των στοιχείων του πίνακα, δε μπορεί να χρησιμοποιηθεί για τροποποίηση των στοιχείων του

Παράδειγμα (4)

Δημιουργία ενός ArrayList τύπου Car («δικού μας τύπου») με τρία αντικείμενα. Με τη χρήση της ιδιότητας της αναφοράς (reference), περνάμε τα αντικείμενα αυτά ως ορίσματα σε μεθόδους αντικειμένων της κλάσης Driver.

```
package exampleArrayList2;

public class Car {
private String brand;


public Car(String brand) {
this.brand = brand;
}



public String getBrand() {
return brand;
}

public void setBrand(String brand) {
this.brand = brand;
}


}
```

```
package exampleArrayList2;

public class Driver {

private String name;
private int age;
private  Car mycar;//αναφορά προς αυτοκίνητο

public Driver(String name, int age) {
this.name = name;
this.age = age;
}

public void setMyCar(Car c) {
mycar = c;
}

public String getName() {
return name;
}
```

```
public int getAge() {
return age;
}

public void printData(){
  System.out.println("Name: "+name);
  System.out.println("age: "+age);
  System.out.println("drives :"
  +mycar.getBrand());
  }

}
```

Παράδειγμα (4)

```
package exampleArrayList2;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Car> cars = new ArrayList<Car>();
Driver D1 = new Driver("Nikos",34); //Δημιουργία αντικειμένου Driver
Driver D2 = new Driver("Kostas",18);
Driver D3 = new Driver("Makis",42);

Car C1 = new Car("Ford");//Δημιουργία αντικειμένου Car
Car C2 = new Car("Fiat");
Car C3 = new Car("Citroen");

D1.setMyCar(C1);//σύνδεση Driver D1->Car C1
D2.setMyCar(C2);//σύνδεση Driver D2->Car C2
D3.setMyCar(C3);//σύνδεση Driver D3->Car C3

cars.add(C1); //προσθήκη αντικειμένου C1 στο cars
cars.add(C2);//προσθήκη αντικειμένου C2 στο cars
cars.add(C3);//προσθήκη αντικειμένου C3 στο cars

//βελτιωμένη πρόταση for
for (Car car : cars) {
      System.out.println(car.getBrand());
    }

//Μέθοδος εκτύπωσης στοιχείων
D1.printData();
D2.printData();
D3.printData();
} }
```

Output:

```
Ford
Fiat
Citroen
Name: Nikos
age: 34
drives :Ford
Name: Kostas
age: 18
drives :Fiat
Name: Makis
age: 42
drives :Citroen
```

Κληρονομικότητα

- Η κληρονομικότητα είναι κεντρική έννοια στον αντικειμενοστρεφή προγραμματισμό.
- Η ιδέα είναι να ορίσουμε μια γενική κλάση που έχει κάποια χαρακτηριστικά (πεδία και μεθόδους) που θέλουμε και μετά να ορίσουμε εξειδικευμένες παραλλαγές της κλάσης αυτής στις οποίες προσθέτουμε ειδικότερα χαρακτηριστικά.
  - Οι εξειδικευμένες κλάσεις λέμε ότι κληρονομούν τα χαρακτηριστικά της γενικής κλάσης

Απλή κληρονομικότητα

- Μηχανισμός υλοποίησης των σχέσεων γενίκευσης/εξειδίκευσης μεταξύ κλάσεων
- Η σχέση εξειδίκευσης «υποκλάση-της» (subclass-of) είναι γνωστή σαν σχέση «είναι ένα» (isa) ή «είναι ένα είδος»
- Μια κλάση είναι υποκλάση μιας μόνο κλάσης
- Η κλάση (υποκλάση) κληρονομεί μεταβλητές και μεθόδους από την (άμεση) υπερκλάση της και τις (έμμεσες) υπερκλάσεις αυτής
- Πλεονέκτημα: αύξηση επαναχρησιμοποίησης

Δημιουργία υποκλάσεων προγραμματιστικά

- Για να επεκτείνουμε μια κλάση γράφουμε την δεσμευμένη λέξη extends και το όνομα της κλάσης που θέλουμε να κληρονομήσουμε.
- Έστω ότι έχουμε την κλάση Vehicle που αντιπροσωπεύει το όχημα και την κλάση Car (αυτοκίνητο). Για να κληρονομίσει η Car την Vehicle γράφουμε:

```
public class Car extends Vehicle
```

Παράδειγμα (5)

Δημιουργούμε μία κλάση Vehicle με υποκλάσεις την Car και Motorcycle.

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

public void prinntInfo_2() { //η εκτύπωση ενναλακτικά
 System.out.println("Τα κυβικά είναι :"+super.getCubicCentimetres());
System.out.println(Το αυτοκίνητο είναι "+brand +" και κινείται
με:"+gasoline);
}

}
```

Παράδειγμα (5)

```
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 prinntInfo() {
super.printInfo();//κλήση μεθόδου printInfo της υπερκλάσης
System.out.println(+super.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.prinntInfo();
MC1.prinntInfo();
}

}
```

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
```

Παράδειγμα (6)

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

```java
package inheritance2;

public abstract class Pet {
    private String name;

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

public String getName() {
return name;
}

    public void speak() {

    }
}
```

```java
package inheritance2;

public class  Dog extends Pet {

private String age;

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

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

```java
package inheritance2;

public class Cat extends Pet {

private String age;

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


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

Παράδειγμα (6)

```
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();


}

}
```

Output:

```
Woof Woof!
Miaou Miaou!
```

Άσκηση για παρουσίες μαθήματος

Να δημιουργήσετε μία κλάση Person με δύο πεδία τύπου String, το name για το όνομα του ατόμου και το age για την ηλικία του. Δημιουργήστε τον κατασκευαστή της κλάσης με τα κατάλληλα ορίσματα για να αρχικοποιήσετε τις τιμές (κάνοντας χρήση του this). Να φτιάξετε τις μεθόδους getName και getAge. Στη συνέχεια, να δημιουργήσετε μία κλάση Student, η οποία είναι υποκλάση της Person και έχει επιπλέον ένα πεδίο τύπου String το am για αριθμό μητρώου. Να υλοποιήσετε τον κατάλληλο κατασκευαστή με την super και να φτιάξετε μία μέθοδο εκτύπωσης, την printInfo για την εκτύπωση του ονόματος, της ηλικίας και του αριθμού μητρώου του φοιτητή. Στη main να φτιάξετε ένα αντικείμενο τύπου Student και να εμφανίσετε τα στοιχεία του μέσω της printInfo.