ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κλάσεις και Αντικείμενα

Constructors, equals, toString

Constructors (Δημιουργοί)

- Ο Constructor είναι μια «μέθοδος» η οποία καλείται όταν δημιουργούμε το αντικείμενο χρησιμοποιώντας την new.
- Αν δεν έχουμε ορίσει Constructor καλείται ένας default constructor χωρίς ορίσματα που δεν κάνει τίποτα.
- Αν ορίσουμε constructor, τότε καλείται ο constructor που ορίσαμε.

Παράδειγμα

```
class Person
{
    private String name;

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

    public void speak(String s){
        System.out.println(name+":"+s);
    }
}

public class HelloWorld3
{
    public static void main(String[] args){
        Person alice = new Person("Alice");
        alice.speak("Hello World");
    }
}
```

Constructor: μια μέθοδος με το ίδιο όνομα όπως και η κλάση και χωρίς τύπο (ούτε void)

Αρχικοποιεί την μεταβλητή name

Constructor: καλείται όταν δημιουργείται το αντικείμενο με την new και μόνο τότε

```
class Date
{
  private int day;
  private int month;
  private int year;
  private String[] monthNames =
           {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

  public Date(int day, int month, int year)
  {
    if (day <= 0 || day > 31 || month <= 0 || month >12 ){
           return;
    }
    this.day = day;
    this.month = month;
    this.year = year;
  }

  public void printDate()
  {
    System.out.println(day + " " + monthNames[month-1] + " " + year);
  }
}

class DateExample
{
  public static void main(String args[])
  {
    Date myDate = new Date(7,3,2013);
    myDate.printDate();
  }
}
```

Παράδειγμα

```
class Car
{
  private int position=0;
  private int ACCELERATOR = 2;

  public Car(int position){
    this.position = position;
  }

  public void move(int delta){
    position += ACCELERATOR * delta ;
  }
}

class MovingCar8
{
  public static void main(String args[]){
          Car myCar1 = new Car(1);
          Car myCar2 = new Car(-1);
          myCar1.move(-1);
          myCar2.move(1);
  }
}
```

Η εκτέλεση αυτών των αρχικοποιήσεων γίνεται πριν εκτελεστούν οι εντολές στον constructor

Η τελική τιμή του position θα είναι αυτή που δίνεται σαν όρισμα

Υπερφόρτωση (Overloading)

- Η Java μας δίνει τη δυνατότητα να ορίσουμε την πολλές μεθόδους με το ίδιο όνομα μέσω της διαδικασίας της υπερφόρτωσης (overloading)
  - Ορισμός πολλών μεθόδων με το ίδιο όνομα αλλά διαφορετικά ορίσματα, μέσα στην ίδια κλάση.

```
class Car
{
        private int position;

        public Car(int position){
                this.position = position;
        }

        public void move(){
                position ++ ;
        }

        public void move(int delta){
                position += delta ;
        }
}

class MovingCar9
{
        public static void main(String args[]){
          Car myCar = new Car(1);
          myCar.move();
          myCar.move(-1);
        }
}
```

Μετακινεί το όχημα μια θέση μπροστά

Μετακινεί το όχημα μια θέση πίσω

Υπογραφή μεθόδου

- Η υπογραφή μίας μεθόδου είναι το όνομα της και η λίστα με τους τύπους των ορισμάτων της μεθόδου
  - Η Java μπορεί να ξεχωρίσει μεθόδους με διαφορετική υπογραφή.
  - Π.χ., move(), move(int) έχουν διαφορετική υπογραφή

Υπερφόρτωση δημιουργών

```
class Car
{
        private int position;

        public Car(){
                this.position = 0;
        }

        public Car(int position){
                this.position = position;
        }

        public void move(){
                position ++ ;
        }

        public void move(int delta){
                position += delta ;
        }

}

class MovingCar10
{
        public static void main(String args[]){
                Car myCar1 = new Car(1); myCar1.move();
                Car myCar2= new Car(); myCar2.move(-1);
        }
}
```

Υπερφόρτωση - Προσοχή

- Όταν ορίζουμε ένα constructor, ο default constructor παύει να υπάρχει. Πρέπει να τον ορίσουμε μόνοι μας.
- Η υπερφόρτωση γίνεται μόνο ως προς τα ορίσματα, ΌΧΙ ως προς την επιστρεφόμενη τιμή.
- Λόγω της συμβατότητας μεταξύ τύπων μια κλήση μπορεί να ταιριάζει με διάφορες μεθόδους. Καλείται αυτή που ταιριάζει ακριβώς, ή αυτή που είναι πιο κοντά.
- Αν υπάρχει ασάφεια στο ποια συνάρτηση πρέπει να κληθεί θα χτυπήσει ο compiler.

Δυο ειδικές μέθοδοι

- Η Java «περιμένει» να δει τις εξής δύο μεθόδους για κάθε αντικείμενο
  - Τη μέθοδος toString η οποία για ένα αντικείμενο επιστρέφει μία string αναπαράσταση του αντικειμένου.
  - Τη μέθοδο equals η οποία ελέγχει για ισότητα δύο αντικειμένων

- Και οι δύο συναρτήσεις ορίζονται από τον προγραμματιστή
  - Το τι String θα επιστραφεί και τι σημαίνει δύο αντικείμενα να είναι ίσα μπορούν να οριστούν όπως μας βολεύει.

Παράδειγμα

- Στην κλάση Car θέλουμε να προσθεσουμε τις μεθόδους toString και equals
  - Η toString θα επιστρέφει ένα String με τη θέση του αυτοκινήτου
  - Η equals θα ελέγχει αν δύο οχήματα έχουν την ίδια θέση.

toString()

Για να μπορούμε να μετατρέψουμε τον ακέραιο σε String ορίζουμε το position ως Integer (wrapper class)

Η Java περιμένει αυτό το συντακτικό για τον ορισμό της toString

Μετά καλούμε τη συνάρτηση toString() της κλάσης Integer

Χρησιμοποιούμε τις myCar1,myCar2 σαν String. Καλείται η μέθοδος toString() αυτόματα

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

  public Car(int position){
    this.position = position;
  }

  public void move(int delta){
    position += delta ;
  }

  public string toString(){
    return position.toString();
  }
}

class MovingCarToString
{
  public static void main(String args[]){
    Car myCar1 = new Car(1);
    Car myCar2 = new Car(0); myCar2.move(2);
    System.out.println("Car 1 is at " + myCar1 + " and car 2 is at " + myCar2);
  }
}
```

Ισοδύναμο με το:

```
System.out.println("Car 1 is at " + myCar1.toString() + " and car 2 is at " + myCar2.toString());
```

toString()

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

  public Car(int position){
    this.position = position;
  }

  public void move(int delta){
    position += delta ;
  }

  public string toString(){
    return ""+position;
  }
}

class MovingCarToString
{
  public static void main(String args[]){
    Car myCar1 = new Car(1);
    Car myCar2 = new Car(0); myCar2.move(2);
    System.out.println("Car 1 is at " + myCar1 + " and car 2 is at " + myCar2);
  }
}
```

Ένας άλλος τρόπος να μετατρέψουμε ένα int σε String

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

  public Car(int position){
    this.position = position;
  }

  public void move(int delta){
    position += delta ;
  }

  public boolean equals(Car other){
    if (this.position == other.position){
      return true;
    }
    return false;
  }
}

class MovingCarEquals
{
  public static void main(String args[]){
    Car myCar1 = new Car(2);
    Car myCar2 = new Car(0); myCar2.move(2);
    if (myCar1.equals(myCar2)){
       System.out.println("Collision!");
    }
  }
}
```

Η Java περιμένει αυτό το συντακτικό για τον ορισμό της equals

Ένα παράδειγμα αντικειμένου ως παράμετρος συνάρτησης

Αν και το πεδίο position είναι private μπορούμε να το προσπελάσουμε γιατί είμαστε μέσα στην κλάση Car.
Μία κλάση μπορεί να προσπελάσει τα ιδιωτικά μέλη όλων των αντικειμένων της κλάσης

Χρήση της return για έλεγχο ροής

Κλήση της equals στο πρόγραμμα

Αντικείμενα ως ορίσματα

- Μπορούμε να περνάμε αντικείμενα ως ορίσματα σε μία μέθοδο όπως οποιαδήποτε άλλη μεταβλητή
- Οποιαδήποτε κλάση μπορεί να χρησιμοποιηθεί ως παράμετρος.
- Όταν τα ορίσματα ανήκουν στην κλάση στην οποία ορίζεται η μέθοδος τότε η μέθοδος μπορεί να δει (και) τα ιδιωτικά (private)πεδία των αντικειμένων
- Αν τα ορίσματα είναι διαφορετικού τύπου τότε η μέθοδος μπορεί μόνο να καλέσει τις public μεθόδους.

Παραδειγμα

- Ορίστε μια μέθοδο που να μας επιστρέφει την απόσταση μεταξύ δύο οχημάτων.

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

  public Car(int position){
    this.position = position;
  }

  public void move(int delta){
    position += delta ;
  }

  public int distanceFrom(Car other){
    return this.position - other.position;
  }
}

class MovingCarDistance
{
  public static void main(String args[]){
    Car myCar1 = new Car(1);
    Car myCar2 = new Car(0); myCar2.move(2);
    System.out.println("Distance of Car 1 from Car 2: " + myCar1.distanceFrom(myCar2));
    System.out.println("Distance of Car 2 from Car 1: " + myCar2.distanceFrom(myCar1));
}
```

Παράδειγμα

- Η κλάση Car θα έχει ως πεδίο και το όνομα του οδηγού. Το όνομα θα το παίρνει από μία ένα αντικείμενο της κλάσης Person στην αρχικοποίηση.

```
class Person
{
  private String name;

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

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

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

  public Car(int position, Person driver){
    this.position = position;
    driverName = driver.getName();
  }

  public String toString(){
    return driverName + " " + position;
  }
}
```

```
class MovingCarDriver
{
  public static void main(String args[])
  {
    Person alice = new Person("Alice");
    Car myCar = new Car(1, alice);
    System.out.println(myCar);
  }
}
```

Αντικείμενα μέσα σε αντικείμενα

- Εκτός από ορίσματα σε μεθόδους αντικείμενα οποιαδήποτε κλάσης μπορούν να εμφανιστούν και ως πεδία μιας κλάσης
  - Ένα αντικείμενο μπορεί να έχει μέσα του άλλα αντικείμενα.

```
class Person
{
  private String name;

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

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

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

  public Car(int position, Person driver){
    this.position = position;
    this.driver = driver;
  }

  public String toString(){
    return driver.getName()
                + " " + position;
  }
}
```

```
class MovingCarDriver
{
  public static void main(String args[])
  {
    Person alice = new Person("Alice");
    Car myCar = new Car(1, alice);
    System.out.println(myCar);
  }
}
```

Κώδικας σε πολλά αρχεία

- Όταν έχουμε πολλές κλάσεις βολεύει να τις βάζουμε σε διαφορετικά αρχεία.
  - Το κάθε αρχείο έχει το όνομα της κλάσης
  - Σημείωση: μια κλάση μόνη της σε ένα αρχείο είναι by default public, μαζι με άλλη είναι by default private.

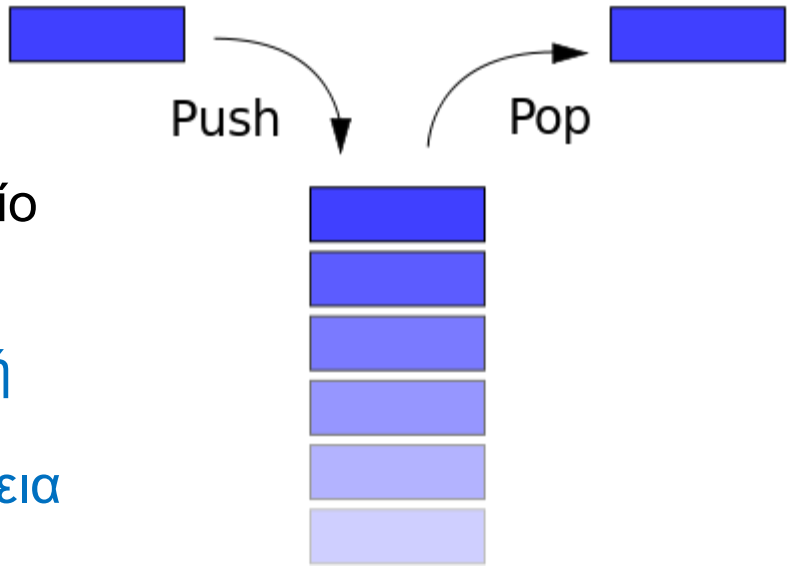
- Για να κάνουμε compile πολλά αρχεία μαζι:
  - `javac file1.java file2.java file3.java`

Αφηρημένοι Τύποι Δεδομένων

- Ο Αφηρημένος Τύπος Δεδομένων (Abstract Data Type – ADT) είναι μια αφαίρεση που περιγράφει μια συλλογή δεδομένων και λειτουργίες που μπορούμε να επιτελέσουμε πάνω σε αυτά τα δεδομένα
  - Ομαδοποιεί Δομές Δεδομένων που υλοποιούν τον Αφηρημένο τύπο Δεδομένων με διαφορετικούς τρόπους.

Παράδειγμα ADT: Στοίβα (Stack)

- Η Στοίβα είναι μια συλλογή δεδομένων η οποία επιτρέπει τις εξής λειτουργίες:
  - push(element): προσθέτει ένα νέο στοιχείο στην κορυφή της στοίβας
  - pop(): αφαιρεί και επιστρέφει το στοιχείο το οποίο βρίσκεται στην κορυφή της στοίβας.
  - top(): διαβάζει το στοιχείο στην κορυφή της στοίβας (δεν το αφαιρεί)
  - isEmpty(): ελέγχει αν η στοίβα είναι άδεια και επιστρέφει true ή false
- Η Στοίβα υλοποιεί την πολιτική Last-In-First-Out (LIFO) στη σειρά που μας δίνει τα στοιχεία
  - Χρήσιμο σε διάφορες εφαρμογές, π.χ., για τη δέσμευση μνήμης στην κλήση συναρτήσεων



Υλοποίηση

- Θα υλοποιήσουμε μια Στοίβα ακεραίων χρησιμοποιώντας ένα πίνακα (Στοιβα συγκεκριμένης χωρητικότητας)
  - Τι πεδία πρέπει να ορίσουμε?
  - Τι μεθόδους?

Εφαρμογές

- Διαβάστε 4 ακεραίους και τυπώστε τους στην αντίθετη σειρά
- Υπολόγισε την δυαδική μορφή ενός ακεραίου.
- Υπολογίστε την συνάρτηση:

$f(x) = 2f(x-1) + 2x + 1,\ f(0) = 1,$

για x=5

Επεκτάσεις

- Πως θα ορίσουμε την μέθοδο equals?
- Πως θα ορίσουμε τη μέθοδο toString?