ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Δημιουργία Κλάσεων και Αντικειμένων

Constructors

Κλάσεις και αντικείμενα

- Ορισμός κλάσης:

```
class <Όνομα Κλάσης>
{
        <Ορισμός πεδίων κλάσης>

        <Ορισμός μεθόδων κλάσης>
}
```

- Ορισμός αντικειμένου:

```
<Όνομα Κλάσης> myObject = new <Όνομα Κλάσης>();
```

  - Ο ορισμός του αντικειμένου γίνεται συνήθως μέσα στη main ή μέσα στη μέθοδο μίας άλλης κλάσης που χρησιμοποιεί το αντικείμενο

Accessor and Mutator methods

- Πολλές φορές χρειαζόμαστε να διαβάσουμε ή να αλλάξουμε ένα πεδίο ενός αντικειμένου
  - Π.χ., να διαβάσουμε τη θέση του οχήματος, ή να τοποθετήσουμε το όχημα σε μια συγκεκριμένη θέση.
  - Πως θα το κάνουμε αφού τα πεδία είναι private?
- Ορίζουμε ειδικές μεθόδους
  - Μέθοδος προσπέλασης (accessor method) για διάβασμα
  - Μέθοδος μεταλλαγής (mutator method) για γράψιμο
- Σύμβαση: Στη Java η ονοματολογία των μεθόδων αυτών γίνεται με συγκεκριμένο τρόπο:
  - get<ονομα μεταβλητης> για την πρόσβαση
    - getPosition()
  - set<ονομα μεταβλητης> για την μετάλλαξη
    - setPosition(<τιμή>)

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

  public void setPosition(int p){
    position = p;
  }

  public int getPosition(){
    return position;
  }

  public void move(){
    position ++ ;
  }
}

class MovingCar7
{
  public static void main(String args[]){
    Car myCar = new Car();
    myCar.setPosition(10);
    myCar.move();
    System.out.println(myCar.getPosition());
  }
}
```

Υπάρχουν περιπτώσεις που μπορεί να θέλουμε η συνάρτηση set να επιστρέφει boolean (true αν η ανάθεση έγινε επιτυχώς, false αλλιώς)

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

        public boolean setPosition(int p){
                if (p < 0){
                        return false;
                }
                position = p;
                return true;
        }

        public int getPosition(){
                return position;
        }

        public void move(){
                position ++ ;
        }
}

class MovingCar7b
{
        public static void main(String args[]){
                Car myCar = new Car();
                boolean check = myCar.setPosition(-1);
                if (!check){
                        System.out.println("position not set");
                }
        }
}
```

Η setPosition μπορεί να επιστρέφει τιμή
Το πιο συνηθισμένο είναι να επιστρέφει boolean αν έγινε σωστά η ανάθεση

Το αντικείμενο this

- Με την δεσμευμένη λέξη this ένα αντικείμενο αναφέρεται στον εαυτό του.

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

    public void setPositionToTen(){
       this.position = 10;
    }
}

class Example
{
   public static void main(String args[]){
       Car myCar = new Car();
       myCar.setPositionToTen();
   }
}
```

Το this.position αναφέρεται στο εσωτερικό πεδίο position του αντικείμενου που θα καλέσει την μέθοδο.

Όταν δημιουργηθεί το αντικείμενο myCar και καλέσει την μέθοδο setPositionToTen το this αναφέρεται πλέον στο αντικείμενο που κάλεσε την μέθοδο, δηλαδή το myCar

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

  public void setPosition(int position){
    this.position = position;
  }

  public int getPosition(){
    return position;
  }

  public void move(){
    position ++ ;
  }
}

class MovingCar8
{
  public static void main(String args[]){
    Car myCar = new Car();
    myCar.setPosition(10);
    myCar.move();
    System.out.println(myCar.getPosition());
  }
}
```

Το this.position αναφέρεται στο πεδίο του αντικειμένου.
Το position αναφέρεται στην παράμετρο της συνάρτησης

Το κρυφό πεδίο this προσδιορίζει το αντικείμενο που κάλεσε την μέθοδο

Έτσι μπορούμε να χρησιμοποιήσουμε το ίδιο όνομα μεταβλητής χωρίς να δημιουργείται σύγχυση

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.
  - Ο default constructor απλά εκτελεί τις αρχικοποιήσεις.
- Αν ορίσουμε constructor, τότε καλείται ο constructor που ορίσαμε.

Συντακτικό

- Ο constructor είναι μια μέθοδος:
  - Που έχει το όνομα της κλάσης
  - Ορίζεται πάντα public
  - Δεν έχει τύπο

```
class <Όνομα κλάσης>
{
      <Ορισμός Πεδίων>
     public <Όνομα κλάσης>([ορίσματα])
     {
            [κώδικας];
     }
}
```

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 HelloWorld2
{
    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

Μια συνομιλία

```
class Person
{
    private String name;

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

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

public class Conversation
{
    public static void main(String[] args){
        Person alice = new Person("Alice");
        Person bob = new Person("Bob");
        alice.speak("Hi Bob");
        bob.speak("Hi Alice");
    }
}
```

Παράδειγμα

```
class Car
{
        private int position;

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

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

        public void printPosition(){
                System.out.println("Car is at position "+position);
        }

}

class MovingCar9
{
        public static void main(String args[]){
                Car myCar1 = new Car(1);
                Car myCar2 = new Car(-1);
                myCar1.move(-1); myCar1.printPosition();
                myCar2.move(1); myCar2.printPosition();
        }
}
```

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 ;
   }

   public void printPosition(){
     System.out.println("Car is at position "+position);
   }
}

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

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

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

Παράδειγμα

- Μία κλάση που να αποθηκεύει ημερομηνίες
  - Η κλάση θα παίρνει την ημέρα, μήνα και χρόνο σαν νούμερα (π.χ., 13 3 2014) και θα μπορεί να τυπώνει την ημερομηνία με το όνομα του μήνα (π.χ., 13 Μαρτίου 2014)
  - Στο πρόγραμμα βάλετε μια ημερομηνία και τυπώστε την.

```
class Date
{
        private int day = 1;
        private int month = 1;
        private int year = 2016;
        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(9,3,2016);
                myDate.printDate();
        }
}
```

```
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 (checkDay(day)){ this.day = day;}
    if (checkMonth(month)){ this.month = month;}
    this.year = year;
  }

  private boolean checkDay(int day){
     if (day <= 0 || day > 31 ) {return false;}
     return true;
  }

  private boolean checkMonth(int day){
     if (month <= 0 || month >12) {return false;}
     return true;
  }

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

Ο constructor μπορεί να καλεί και άλλες μεθόδους που κάνουν κάποια από τη δουλειά που χρειάζεται

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

```
class Date
{
        private int day = 1;
        private int month = 1;
        private int year = 2016;
        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(9,3,2016);
                myDate.printDate();
        }
}
```

Αν μπούμε στο if οι τελικές τιμές των ορισμάτων θα είναι αυτές που θα δοθούν στον constructor . Αλλιώς διατηρούνται οι αρχικές τιμές

Παραδείγματα

- Θέλουμε μια κλάση Student που να κρατάει πληροφορίες για έναν φοιτητή. Τι πεδία πρέπει να έχουμε? Τι θα μπει στον constructor?

- Θέλουμε μια κλάση (GuestList) που να χειρίζεται τους καλεσμένους σε ένα πάρτι. Τι πεδία πρέπει να έχουμε? Πώς θα κάνουμε τον constructor?

```
class Student
{
	private String name = "John Doe";
	private int AM = 1000;


	public Student(String name, int AM){
		this.name = name;
		this.AM = AM;
	}

	public void printInfo(){
		System.out.println(name + " " + AM);
	}

	public static void main(String[] args){
		Student aStudent = new Student("Kostas", 1001);
		aStudent.printInfo();
	}
}
```

Guest List

Ορίζει τους πίνακες με τα ονόματα των καλεσμένων και τις επιβεβαιώσεις, αλλά δεν δεσμεύει χώρο γιατί δεν ξέρουμε τον αριθμό των προσκεκλημένων.

Δεσμεύει μνήμη για τους πίνακες με τα ονόματα των καλεσμένων και τις επιβεβαιώσεις

Αρχικοποίηση του αριθμού των επισκεπτών

```
class GuestList
{
        private String[] names;
        private boolean[] confirm;
        int numberOfGuests;

        public GuestList(int numberOfGuests)
        {
                this.numberOfGuests = numberOfGuests;
                names = new String[numberOfGuests];
                confirm = new boolean[numberOfGuests];
                // Εδώ μπορούμε να έχουμε κώδικα για τις τιμές
                // Ή να εισάγονται τα ονόματα ένα ένα.
                // Ή μπορεί η εισαγωγή ονομάτων να γίνει
                // με άλλες μεθόδους της κλάσης
        }

}
```

Μια υλοποίηση με μέθοδο για προσθήκη προσκεκλημένων

```
class GuestList
{
        private String[] names;
        private boolean[] confirm;
        int numberOfGuests;
        int guestsSoFar = 0;

        public GuestList(int numberOfGuests)
        {
                this.numberOfGuests = numberOfGuests;
                names = new String[numberOfGuests];
                confirm = new boolean[numberOfGuests];
        }

        public void addGuest(String name, Boolean confirmation){
        if (guestsSoFar == numberOfGuests){
                return;
        }
                names[guestsSoFar] = name;
                confirm[guestsSoFar] = confirmation;
                guestsSoFar ++;
        }
}
```

Χρειαζόμαστε αυτή τη μεταβλητή για να ξέρουμε πόσους επισκέπτες έχουμε προσθέσει μέχρι τώρα

Αν έχει γεμίσει η λίστα μας δεν προσθέτουμε

Η guestsSoFar μας δίνει και την επόμενη άδεια θέση στον πίνακα.