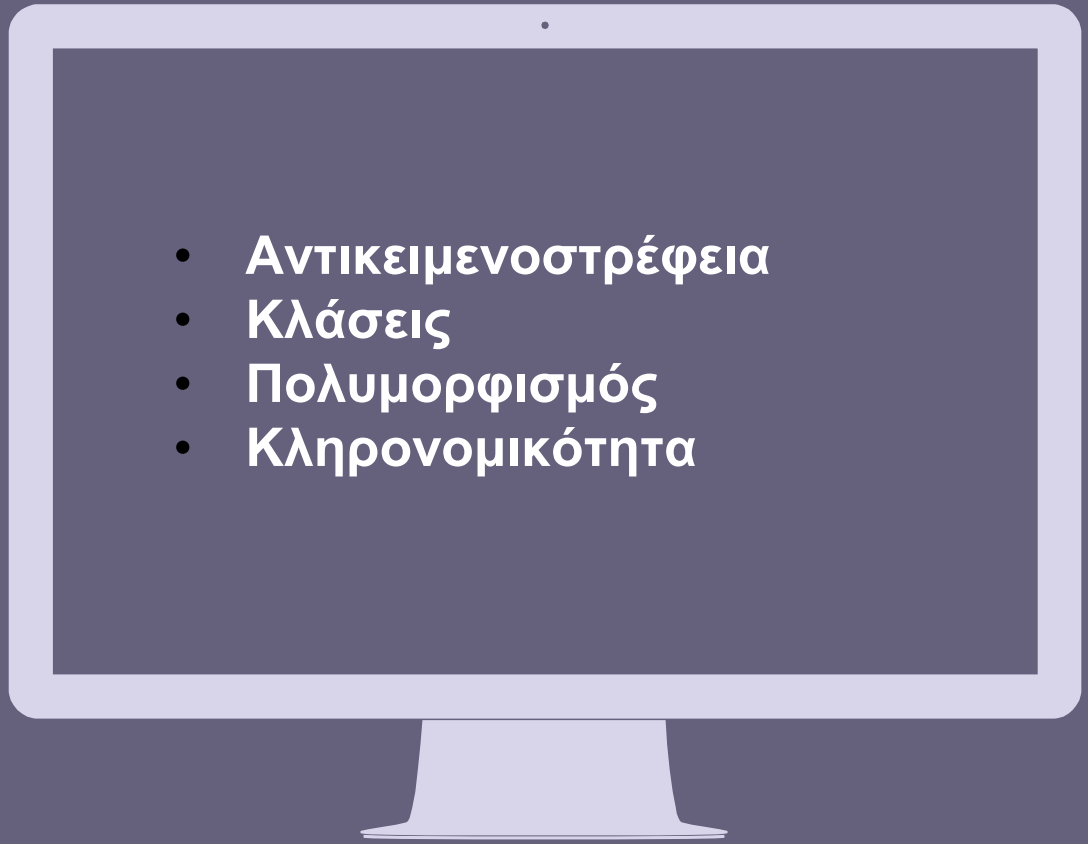
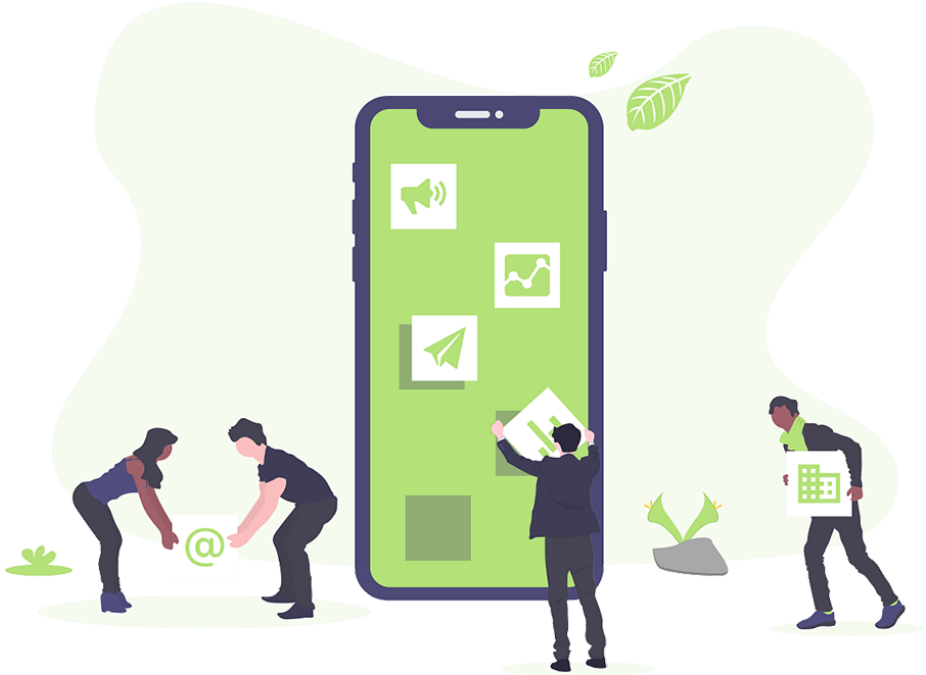
2ο μέρος

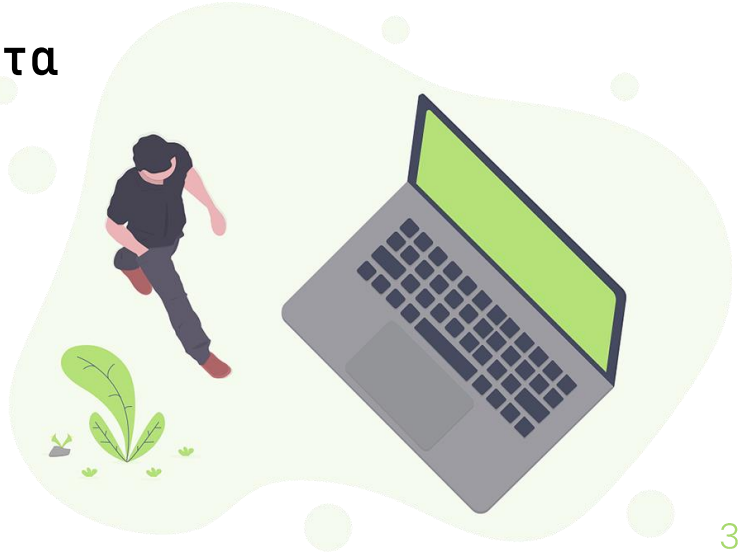

• Αντικειμενοστρέφεια
• Κλάσεις
• Πολυμορφισμός
• Κληρονομικότητα

Αντικειμενοστρέφεια



Τι είναι η αντικειμενοστρέφεια;

Αντικειμενοστρέφεια ή αντικειμενοστρεφής προγραμματισμός ονομάζεται η τεχνική προγραμματισμού που έχει ως 'δομικό' υλικό τα αντικείμενα. Σε αντίθεση με τον κλασικό δομημένο προγραμματισμό ο αντικειμενοστρεφής είναι ευκολότερο να διατηρηθεί και να επεκταθεί

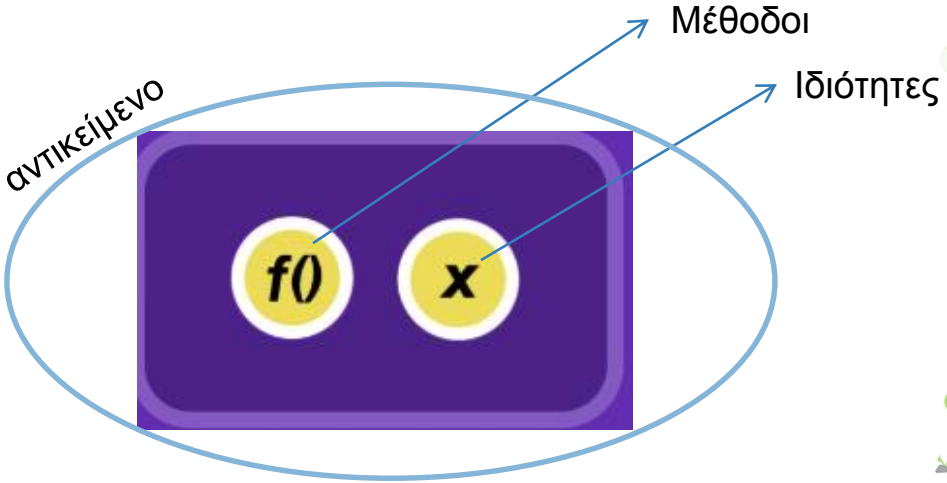


Η ουσία της αντικειμενοστρέφειας;

Η ενοποίηση μεταβλητών και συναρτήσεων που σχετίζονται μεταξύ τους σε ένα unit . Αυτό το unit ονομάζεται Αντικείμενο

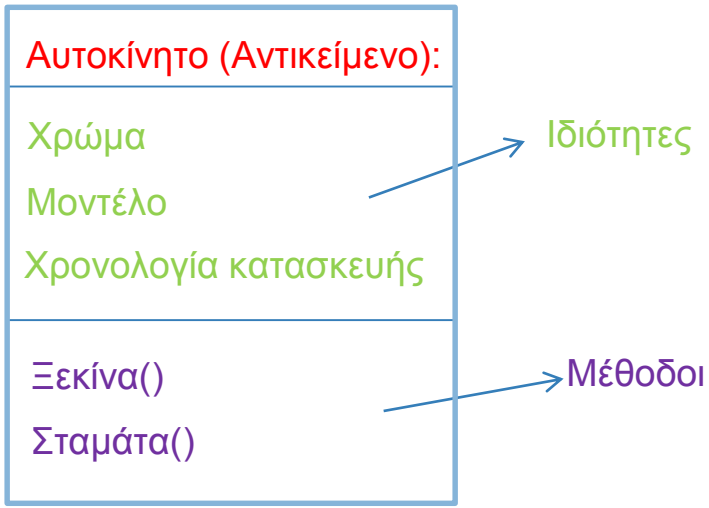
Αντικείμενο:

Οι μεταβλητές που βρίσκονται μέσα σε ένα αντικείμενο ονομάζονται ιδιότητες και οι συναρτήσεις ονομάζονται μέθοδοι



Ένα παράδειγμα αντικειμένου:

Έστω ότι σκεφτόμαστε ένα αυτοκίνητο. Ποιες μπορεί να είναι οι ιδιότητες και ποιες οι μέθοδοι του;



Πως φτιάχνουμε αντικείμενα στην java;

Αντικείμενα στην java φτιάχνουμε με το **class**

```
public class MyClass{

}
```

Ας φτιάξουμε το Προηγούμενο Παράδειγμα με το αυτοκίνητο σε java:

```java
public class Car {
    String model;
    String color;
    int year_made;

    public void start(){
        System.out.println("Starting...");
    }

    public void stop(){
        System.out.println("Stopping...");
    }
}
```

Πως φτιάχνουμε ένα αντικείμενο και Πως του δίνουμε τιμές;

```java
public class Car {
    String model;
    String color;
    int year_made;

    public void start(){
        System.out.println("Starting...");
    }

    public void stop(){
        System.out.println("Stopping...");
    }
}
```

```java
public class Main {
    public static void main(String args[]){
        Car car1 = new Car();
        car1.color = "White";
        car1.year_made = 2014;
        car1.model= "Bmw i8";

        System.out.println(car1.color);

        car1.start();
    }
}
```

Δημιουργία αντικειμένου car

Output

```
White
Starting...
```

Τι είναι ο Constructor;

```java
public class Car {
    String model;
    String color;
    int year_made;

    public Car(String model, String color, int year_made){
        this.model = model;
        this.color = color;
        this.year_made = year_made;
    }

    public void start(){
        System.out.println("Starting...");
    }

    public void stop(){
        System.out.println("Stopping...");
    }
}
```

Constructor - "κατασκευάζει" - "γεμίζει" με τιμές το αντικείμενο μας κατά την δημιουργία του

Η δεσμευμένει λέξη this χρησιμοποιείται για να αναφερθούμε σε πεδία της class

Δίνουμε τιμές στον Constructor

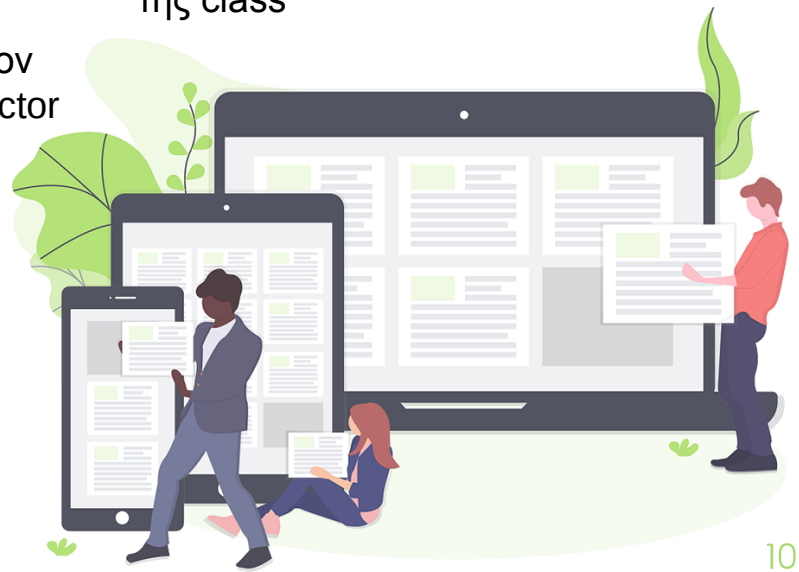
```java
public class Main {
    public static void main(String args[]){
        Car car1 = new Car("BMW i8", "White", 2004);
        System.out.println(car1.color);

        car1.start();

        Car car2 = new Car();

    }
}
```

Γιατί error;



Return types:

```
public class Operations {
    public int sum(int x, int y){
        return x + y;
    }
}
```

```
public class Main {
    public static void main(String args[]){
        Operations op = new Operations();

        System.out.println(op.sum(2, 2));
    }
}
```

Method Overloading

```java
public class Operations {
    public int sum(int x, int y){
        return x + y;
    }

    public double sum(double x, double y){
        return x + y;
    }
}
```

```java
public class Main {
    public static void main(String args[]){
        Operations op = new Operations();

        System.out.println(op.sum(2, 2));
        System.out.println(op.sum(2.4, 2.3));
    }
}
```

Output:

```
run:
4
4.699999999999999
BUILD SUCCESSFUL (total time: 0 seconds)
```

Γιατί;

Public – Έχουν πρόσβασή όλες οι κλάσεις

Protected – Έχουν πρόσβαση, οι κλάσεις στο ίδιο πακέτο και οι υπο κλάσεις

Private - Έχει πρόσβαση μονο η κλάση

Default (αν δεν βάλουμε τίποτα) – Έχουν πρόσβαση μόνο αυτές που είναι στο ίδιο πακέτο

Παράδειγματα:

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}

public class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);//Compile Time Error
   obj.msg();//Compile Time Error
   }
}
```

Παράδειγματα:

```java
//save by A.java
package pack;
class A{
  void msg(){System.out.println("Hello");}
}
```

```java
//save by B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();//Compile Time Error
   obj.msg();//Compile Time Error
  }
}
```

Παράδειγματα:

```java
//save by A.java
package pack;
public class A{
protected void msg(){System.out.println("Hello");}
}
```

```java
//save by B.java
package mypack;
import pack.*;

class B extends A{
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
  }
}
```

```
Output:Hello
```

Παράδειγματα:

```java
//save by A.java

package pack;
public class A{
public void msg(){System.out.println("Hello");}
}
```

```java
//save by B.java

package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

```
Output:Hello
```

Pass-by-reference στην Java;

Όλα στην java είναι pass-by-value

```
public class Main {

    public static void main(String args[]){
        Dog myDog = new Dog("Mac");
        foo(myDog);
    }

    public static void foo(Dog dog){
        dog.setName("Max");
        dog = new Dog("Jack"); // A new Dog object
        dog.setName("Jackson");
    }
}
```

Τι όνομα έχει το αντικείμενο myDog τελικά;

Static Member variable:

```
public class Dog {
    public String name;
    public static int numberOfDogs = 0;

    public Dog(String name){
        this.name = name;
        numberOfDogs++;
    }

    public void setName(String name){
        this.name = name;
    }
}
```

Static class:

```
public class Project {
    public class A{
        public void say(){
            System.out.println("Hey!");
        }
    }

    public static class Dog{
        public String name;
        public static int numberOfDogs = 0;

        public Dog(String name){
            this.name = name;
            numberOfDogs++;
        }

        public void setName(String name){
            this.name = name;
        }
    }
}
```

Top class δεν μπορεί να είναι static

Non-static nested ή inner class

Static nested class

Static vs non-static

Static:

- Δεν χρειάζονται reference του outer class
- Μπορεί να έχει πρόσβαση μόνο σε static πεδία & μεθόδους

Non-static:

- Χρειάζονται reference του outer class
- Μπορεί να έχει πρόσβαση σε static και non-static πεδία & μεθόδους

Παράδειγμα:

```
class OuterClass{
   private static String msg = "Hello";
   public static class NestedStaticClass{
       public void printMessage() {

         System.out.println("Message from nested static class: " + msg);
       }
    }


    public class InnerClass{
       public void display(){
         System.out.println("Message from non-static nested class: "+ msg);
       }
    }
}
class Main
{

   public static void main(String args[]){
      OuterClass.NestedStaticClass printer = new OuterClass.NestedStaticClass();
      printer.printMessage();

      OuterClass outer = new OuterClass();
      OuterClass.InnerClass inner  = outer.new InnerClass();
      inner.display();
      OuterClass.InnerClass innerObject = new OuterClass().new InnerClass();

      innerObject.display();
   }
}
```

Output:

```
Message from nested static class: Hello
Message from non-static nested class: Hello
Message from non-static nested class: Hello
```

Κληρονομικότητα

Είναι η δυνατότητα μια class να πάρει τις ιδιότητες και τις μεθόδους μιας άλλης class

κληρονομικότητας:

Άν και διαφορετικά ζώα έχουν κάποια κοινά. Ποια είναι;

Θα φτιάξουμε Dog και Cat class που κληρονομούν από μια Animal class

Super class ή πατρική class

```
public class Animal {
    int size;
    String color;
    String sound;

    public Animal(int size, String color, String sound){
        this.size = size;
        this.color = color;
        this.sound = sound;
    }
    public void sound(){
        System.out.println(this.sound);
    }
    public void sleep(){
        System.out.println("Sleeping...");
    }
    public void eat(){
        System.out.println("Eating...");
    }
}
```

Δεσμευμένη λέξη

```
public class Dog extends Animal{
    public String name;
    public Dog(String name, String color, int size){
        super(size, color, "Woof");
        this.name = name;
    }
}
```

καλεί τον constructor της πατρικής κλάσης

Προσθέτω έξτρα μεταβλητές στο παιδί

```
public class Cat extends Animal{
    public String name;
    public Cat(String name, String color, int size){
        super(size, color, "Meow");
        this.name = name;
    }
}
```

Παράδειγμα κληρονομικότητας

```
public static void main(String args[]){
    Dog dog = new Dog("Conan", "White", 10);
    Cat cat = new Cat("Mona", "Brown", 10);

    dog.sound();
    cat.sound();
}
```

Πολυμορφισμός

Ο Πολυμορφισμός συμβαίνει όταν έχουμε πολλές κλάσεις που κληρονομούν από μια άλλη. Ο πολυμορφισμός χρησιμοποιεί αυτές τις μεθόδους για την εκτέλεση διαφορετικών εργασιών. Αυτό μας επιτρέπει να εκτελέσουμε μια εργασία με διάφορους τρόπους.

Πολυμορφισμός

```java
public class Animal {
    public Animal(){}

    public void sound(){
        System.out.println("Animal sound");
    }
}
```

```java
public class Dog extends Animal{
    public String name;
    public Dog(String name){
        this.name = name;
    }

    public void sound(){
        System.out.println("Woof");
    }
}
```

Abstract class:

```java
public abstract class Animal {
    int size;
    String color;

    public Animal(int size, String color){
        this.size = size;
        this.color = color;
    }

    abstract public void sound();
}
```

Δεν μπορώ να φτιάξω αντικείμενο Animal μπορώ μόνο να το κάνω extend

```java
public class Dog extends Animal{
    public String name;
    public Dog(String name, String color, int size){
        super(size, color);
        this.name = name;
    }

    public void sound(){
        System.out.println("Woof");
    }
}
```

Πρέπει να υλοποιήσω όλες τις abstract μεθόδους

Special methods:

Γιατι να μην κάνω αντικείμενο == αντικείμενο;

Είναι μέθοδοι οι οποίες κατέχουν όλα τα αντικείμενα στην java: equals(), toString()

```
public class Dog extends Animal{
    public String name;
    public Dog(String name, String color, int size){
        super(size, color);
        this.name = name;
    }

    public void sound(){
        System.out.println("Woof");
    }

    public boolean equals(Dog obj){
        return obj.name.equals(this.name) &&
               obj.size == this.size &&
               obj.color.equals(this.color);
    }

    public String toString(){
        return "A dog named: " + this.name;
    }
}
```

Thanks!

Ερωτήσεις;