ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αντικείμενα μέσα σε αντικείμενα

Αντικείμενα ως επιστρεφόμενες τιμές

Αντικείμενα με πίνακες

ΑΝΤΙΚΕΙΜΕΝΑ ΜΕΣΑ ΣΕ ΑΝΤΙΚΕΙΜΕΝΑ

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, String name){
    this.position = position;
    this.driver = new Person(name);
  }

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

```
class MovingCarDriver1
{
  public static void main(String args[])
  {
    Car myCar = new Car(1, "Alice");
    System.out.println(myCar);
  }
}
```

Το αντικείμενο δημιουργείται μέσα στον constructor
Αυτό έχει νόημα αν το Person χρησιμοποιείται μόνο μέσα στην κλάση Car

```
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 MovingCarDriver2
{
  public static void main(String args[])
  {
    Person alice = new Person("Alice");
    Car myCar = new Car(1, alice);
    System.out.println(myCar);
  }
}
```

Καλύτερη υλοποίηση!

```
class Person
{
  private String name;
  private int age;

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

  public String getName(){
    return name;
  }

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

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

  public Car(int position, Person driver){
    this.position = position;
    if (driver.getAge() >= 18){
        this.driver = driver;
    }
  }

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

```
class MovingCarDriver3
{
  public static void main(String args[])
  {
    Person alice = new Person("Alice");
    Car myCar = new Car(1, alice);
    System.out.println(myCar);
  }
}
```

Η Person είναι διαφορετική κλάση άρα δεν μπορούμε να διαβάσουμε το πεδίο age

Η εντολή exit

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

   public Car(int position, Person driver){
      this.position = position;
      if (driver.getAge() >= 18){
         this.driver = driver;
      }
      else{
         System.exit(-1);
      }
   }
}
```

Χρησιμοποιείται για σοβαρά λάθη για να σταματάει την εκτέλεση του προγράμματος.

Αν δώσουμε μη αποδεκτή ηλικία το πρόγραμμα μας θα σταματήσει.

Το -1 εξυπηρετεί σαν κωδικός λάθους, μπορείτε να βάλετε όποια τιμή θέλετε.

```
class Person
{
  private String name;
  private int licence;

  public Person(String name,
                int licence){
      this.name = name;
      this.licence = licence;
  }
}
```

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

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

Πως θα υλοποιήσουμε την toString και την equals?

```
class Person
{
  private String name;
  private int licence;

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

  public String toString(){
    return name + " " + licence;
  }

  public boolean equals(Person other){
    if (this.name.equals(other.name)&&
      this.licence == other.licence)){
        return true
    }else{
        return false;
    }
  }
}
```

```
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 + " " + position;
  }

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

Φωλιασμένη κλήση της toString και της equals

Φωλιασμένες toString και equals

- Ένα αντικείμενο μπορεί να περιέχει μέσα άλλα αντικείμενα.
- Είναι συνηθισμένο σε αυτή την περίπτωση η toString και equals να ορίζονται κάνοντας φωλιασμένη κλήση της toString και της equals των αντικειμένων που περιέχει.
- Φωλιασμένη κλήση της toString της κλάσης driver

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

- Φωλιασμένη κλήση της toString της κλάσης driver

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

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, μαζί με άλλη είναι by default private.
- Ένα επιπλέον πλεονέκτημα είναι ότι μπορούμε να ορίσουμε μια main συνάρτηση για κάθε κλάση ξεχωριστά
  - Βοηθάει για το testing του κώδικα.

- Για να κάνουμε compile πολλά αρχεία μαζι:
  - `javac file1.java file2.java file3.java`
  - ή μπορούμε να κάνουμε compile το “βασικό” αρχείο

ΑΝΤΙΚΕΙΜΕΝΑ ΩΣ ΕΠΙΣΤΡΕΦΟΜΕΝΕΣ ΤΙΜΕΣ

Αντικείμενα ως επιστρεφόμενες τιμές

- Μία μέθοδος μπορεί να επιστρέφει αντικείμενα όπως οποιαδήποτε άλλη τιμή.

- Είναι δυνατόν επίσης μέσα σε μία μέθοδο να δημιουργούμε ένα αντικείμενο και να το επιστρέφουμε για να χρησιμοποιηθεί μετά.

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

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

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

  public Person getDriver(){
    return driver;
  }
}
```

Επιστρέφει το αντικείμενο Person το οποίο είναι ο οδηγός του οχήματος.

Παράδειγμα

- Στην κλάση BankAccount που κάναμε στο προηγούμενο μάθημα, κάνετε την μέθοδο merge να επιστρέφει ένα καινούριο λογαριασμό

```
class BankAccount
{
    private String name;
    private int amount;

    public BankAccount(String name, int amount){
        this.name = name;
        this.amount = amount;
    }

    public void merge(BankAccount other){
        if (this.name.equals(other.name)){
            this.amount += other.amount;
        }
    }

    public BankAccount mergeIntoNewAccount(BankAccount other){
        if (this.name.equals(other.name)){
            BankAccount newAccount =
                        new BankAccount(name,this.amount+other.amount);
            return newAccount;
        }
        return null;
    }
}
```

Μια άλλη επιλογή είναι να δημιουργήσουμε ένα νέο λογαριασμό μετά την συγχώνευση

Δημιουργούμε ένα νέο αντικείμενο BankAccount και το επιστρέφουμε.

Αν δεν μπορούμε να δημιουργήσουμε το νέο λογαριασμό επιστρέφουμε null. Το null είναι το κενό αντικείμενο.

Παράδειγμα

- Κάνετε μια μέθοδο για την κλάση Car η οποία να παίρνει σαν όρισμα ένα αριθμό Κ και να κάνει Κ κινήσεις με τυχαίο αριθμό από βήματα στο διάστημα μεταξύ 0 έως 9

```
import java.util.Random;

class Car
{
	private int position = 0;

	public int[] move(int K){
		int[] moves = new int[K];
		Random rnd = new Random();
		for (int i = 0; i < K; i ++){
			int delta = rnd.nextInt(10);
			position += delta ;
			moves[i] = delta;
		}
		return moves;
	}

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

}

class MovingCar14
{
	public static void main(String args[]){
		Car myCar = new Car();
		int[] moves = myCar.move(4);
		myCar.printPosition();
		System.out.print("moves: ");
		for (int i = 0; i < 4; i ++){
			System.out.print(moves[i]+" ");
		}
		System.out.println();
	}
}
```

Παράδειγμα

- Στην κλάση DynamicArray φτιάξετε δύο μεθόδους, μία που να χειρίζεται την περίπτωση που χρειάζεται να διπλασιάσουμε τον πίνακα και μία που πρέπει να τον υποδιπλασιάσουμε. Οι μέθοδοι θα επιστρέφουν ένα νέο πίνακα.

```
class DynamicArray
{
    private int[] doubleCapacity()
    {
         capacity = 2*capacity;
         int[] tempArray = new int[capacity];
         for (int i = 0; i < size; i ++){
        tempArray[i] = array[i];
         }
         return tempArray;
    }

     private int[] halfCapacity()
     {
        capacity = capacity/2;
        int[] tempArray = new int[capacity];
        for (int i = 0; i < size; i ++){
         tempArray[i] = array[i];
        }
        return tempArray;
     }
}
```

Η μέθοδος επιστρέφει τον πίνακα και άρα ο χώρος διατηρείται και μετά το τέλος της μεθόδου

```
class DynamicArray
{
    public void add(int x)
    {
        if (size == capacity){
         array = doubleCapacity();
        }
        array[size] = x;
        size ++;
    }

    public int remove()
    {
         if (size == 0){
        return -1;
         }
         size -- ;
         int retValue = array[size];
         if (size == capacity/4){
        array = halfCapacity();
         }
         return retValue;
    }

}
```

ΑΝΤΙΚΕΙΜΕΝΑ ΜΕ ΠΙΝΑΚΕΣ ΙΙ

Παράδειγμα

- Δημιουργήστε μια κλάση CarWithPassengers η οποία μπορεί να κρατάει στο όχημα πολλαπλούς επιβάτες, αντικείμενα Person.

Η CAR_SIZE είναι σταθερά για την κλάση

Δημιουργία πίνακα από αντικείμενα Person

toString με φωλιασμένες κλήσεις

Μπορούμε να τεστάρουμε το πρόγραμμα με μία main στην CarWithPassengers

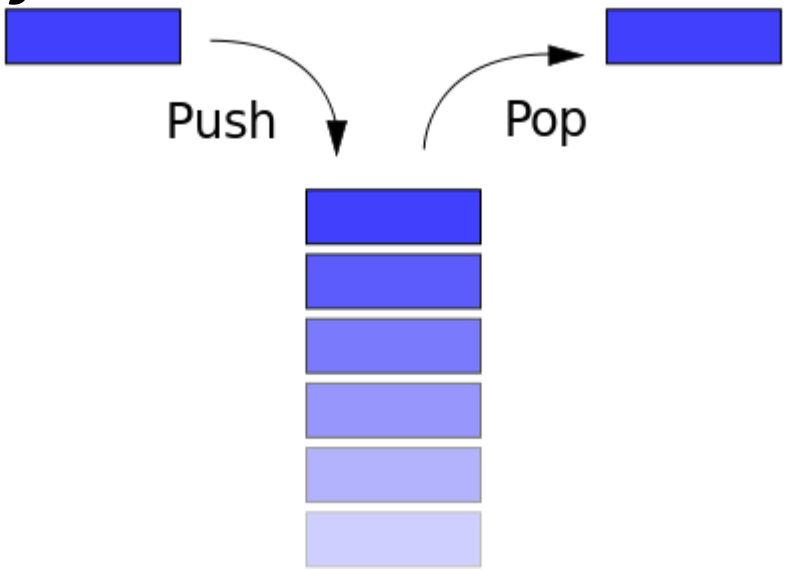
```
class CarWithPassengers
{
    private int CAR_SIZE = 4;
    private int position = 0;
    private Person[] passengers = new Person[CAR_SIZE];
    private int numOfPassengers = 0;

    public void addPassenger(Person passenger)
    {
        if (numOfPassengers < CAR_SIZE){
            passengers[numOfPassengers] = passenger;
            numOfPassengers ++;
            }
    }

    public String toString(){
        String retValue = "Car at "+position +" with passengers: ";
        for (int i = 0; i < numOfPassengers; i ++){
            retValue += passengers[i]+ " ";
        }
        return retValue;
    }
}
```

```
class Person
{
    private String name;

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

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

```
public static void main(String[] args){
    CarWithPassengers myCar = new CarWithPassengers
    Person alice = new Person("Alice");
    myCar.addPassenger(alice);
    Person bob = new Person("Bob");
    myCar.addPassenger(bob);
    System.out.println(myCar);
}
```

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.
  - 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?

```
class Stack
{
        private int capacity;
        private int size = 0;
        private int[] elements;

        public Stack(int capacity){
                this.capacity = capacity;
                elements = new int[capacity];
        }

        public void push(int element){
                if (size == capacity){
                        System.out.println("Cannot enter any more elements");
                        return;
                }
                elements[size] = element;
                size ++;
        }

        public int pop(){
                if (size == 0){
                        System.out.println("No elements to pop");
                        return -1;
                }
                size -- ;
                return elements[size];
        }

        public boolean isEmpty(){
                return (size == 0);
        }
}
```

Επεκτάσεις

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

```
public String toString(){
	String returnString = "":
	for (int i = 0; i < size; i ++){
		returnString = returnString + elements[i] + " ";
	}
	return returnString;
}

public boolean equals(Stack other){
	if (this.size != other.size){
		return false;
	}
	for (int i = 0; i < size; i ++){
		if (this.elements[i] != other.elements[i]){
			return false;
		}
	}
	return true;
}
```