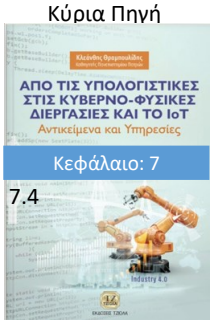
Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming)

(CEID_NNY106)

Διεπαφές
Interfaces

Κύρια Πηγή



Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

```
Interface OperandIf {
   void addDigit(char ch);
   void deleteLastDigit();
   void complete();
   }
```

Java

High-level programming language

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. Wikipedia

Designed by: James Gosling

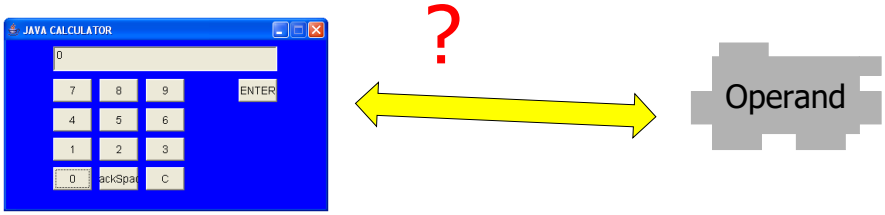
First appeared: May 23, 1995; 27 years ago

Paradigm: Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent

Οργάνωση Διάλεξης

- Το κίνητρο για την κατασκευή του Interface
  - CalcGui - Operand interaction
- Ορισμός του Interface
- Interface inheritance (extends)
- Interface vs Abstract κλάση
- Παραδείγματα interface από την βασική βιβλιοθήκη
  - List
  - Iterator
- Συλλογές (Collections)
- Default και Static μέθοδοι στο interface

CalcGui - Operand interaction




```
CalculatorGui(Operand op);
```

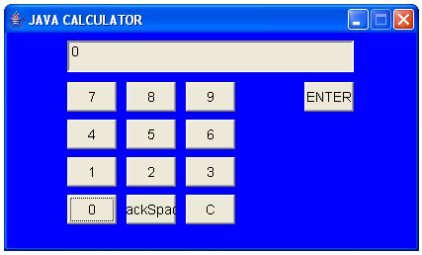
- Κάθε ομάδα προγραμματιστών θα πρέπει να μπορεί να γράψει τον κώδικα της κατασκευής της χωρίς να γνωρίζει πως είναι γραμμένος ο κώδικας της άλλης κατασκευής (*without any knowledge of how the other group's code is written*).
- Αυτό μπορεί να γίνει αν οι δύο ομάδες συμφωνήσουν στο πως οι κατασκευές τους θα επικοινωνούν. Καταλήξουν δηλαδή σε μια συμφωνία (ένα "συμβόλαιο" ("**contract**")) που ορίζει την επικοινωνία.
- ***interfaces* are** such **contracts**.

Interface

- Είναι ένας **τύπος αναφοράς** (reference type) όπως και η κλάση.
- Μπορεί να περιέχει μόνο
  - **constants**,
  - method signatures, and
  - nested types (+ default and static methods @Java8)
- Είναι μια περιγραφή ενός συνόλου χαρακτηριστικών που παρέχονται (**provided**) ή απαιτούνται (**required**) από ένα στιγμιότυπο.
- Μπορεί να θεωρηθεί ως η προβολή (projection) των εμφανών από έξω χαρακτηριστικών μιας οντότητας. Η οντότητα αυτή λέμε ότι **υλοποιεί** (realizes) **το interface**.

Στιγμιότυπα διαφορετικών κλάσεων μπορεί να "παίζουν" τον ίδιο ρόλο, αρκεί ο ρόλος αυτός να ορίζεται από ένα interface, το οποίο **οι κλάσεις υλοποιούν** (**implement**).

CalcGui – Operand interaction



?

```
CalculatorGui(OperandIf op);
```

Interface OperandIf {
  void addDigit(char ch);
  void deleteLastDigit();
  void complete();
  }

class Operand implements OperandIf {
  …
  }

Operand

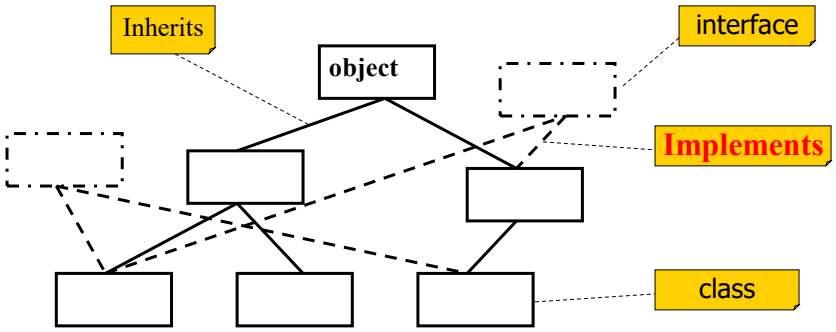
class StackOperand implements OperandIf {
  …
  }

Interface inheritance

```
public interface GroupedInterface extends Interface1, Interface2, Interface3 {
    // constant declarations
    double E = 2.718282;  // base of natural logarithms
    // method signatures
    void doSomething (int i, double x);
    int doSomethingElse(String s);
    }
```

- Ένα interface μπορεί να κληρονομεί (extend) περισσότερα του ενός interfaces
- **Όλοι οι μέθοδοι** που ορίζει ένα interface είναι εξ ορισμού **public**, έτσι ο προσδιοριστής public μπορεί να παραληφθεί.
- **Όλες οι σταθερές** που ορίζει ένα interface είναι εξ ορισμού **public**, **static**, and **final** (οι προσδιοριστές μπορούν να παραληφθούν).

Πολλαπλή Κληρονομικότητα στην Java - Interfaces



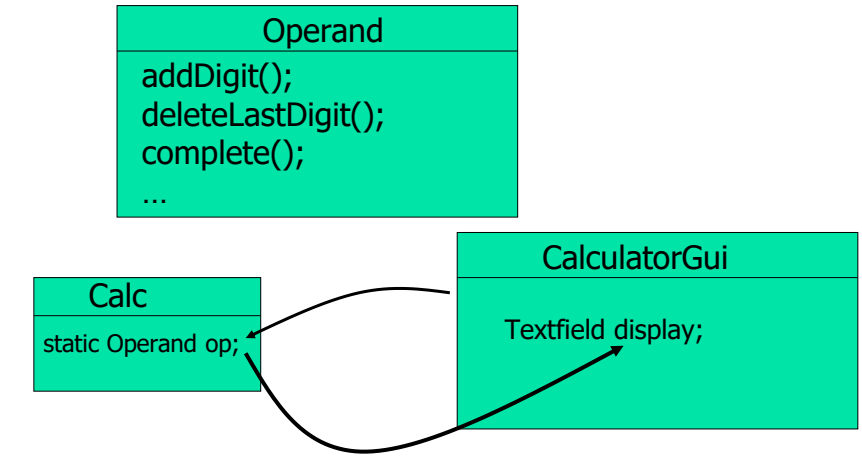
είναι δυνατή η δήλωση μεταβλητής του συγκεκριμένου τύπου interface και η ανάθεση σε αυτή στιγμιότυπου οποιασδήποτε από τις κλάσεις που υλοποιεί το interface

Abstract Classes vs. Interfaces

- Σε αντίθεση με τα interfaces, οι abstract κλάσεις μπορεί να περιέχουν
  - data members που δεν είναι static και final, και
  - υλοποιήσεις μεθόδων.
- Αυτές οι κλάσεις είναι ανάλογες με τα interfaces με την διαφορά ότι ορίζουν ένα μέρος της υλοποίησης.
- Αν μια κλάση περιέχει μόνο **abstract μεθόδους**, είναι προτιμότερο να ορίζεται ως interface.

```
public interface OperatorIf {
        void operate();
}
```

Interfaces in RPN ?



CalculatorGui.display.setText();

Calc.op.addDigit();

Interface List

java.util

Interface List<E>

All Superinterfaces:

`Collection<E>, Iterable<E>`

All Known Implementing Classes:

`AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector`

An ordered collection (also known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that `e1.equals(e2)`, and they typically allow multiple null elements if they allow null elements at all. It is not

The `List` interface provides two methods to search for a specified object. From a performance standpoint, these methods should be used with caution. In many implementations they will perform costly linear searches.

The `List` interface provides two methods to efficiently insert and remove multiple elements at an arbitrary point in the list.

Interface List – Selected Methods

boolean	**add**(E e) Appends the specified element to the end of this list (optional operation).
void	**add**(int index, E element) Inserts the specified element at the specified position in this list (optional operation).
boolean	**contains**(Object o) Returns true if this list contains the specified element.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
Iterator<E>	**iterator**() Returns an iterator over the elements in this list in proper sequence.

Iterator – Iterator Interface

java.util

Interface Iterator<E>

- An Iterator is an object that can be used to loop through collections, like ArrayList, LinkedLists, …
- **iterator()**: method to get an Iterator for any collection

Modifier and Type	Method and Description
default void	**forEachRemaining**(Consumer<? super E> action) Performs the given action for each remaining element until all elements have been processed or the action throws an exception.
boolean	**hasNext**() Returns true if the iteration has more elements.
E	**next**() Returns the next element in the iteration.
default void	**remove**() Removes from the underlying collection the last element returned by this iterator (optional operation).

Java Collections (Συλλογές)

Συλλογή είναι μια ομάδα αντικειμένων, τα οποία αναφέρονται ως στοιχεία (elements) της συλλογής.

- Ένα πλαίσιο εργασίας (framework) που ορίζει μια αρχιτεκτονική για αποθήκευση και διαχείριση μιας συλλογής αντικειμένων.
- Αποτελείται από ένα σύνολο από interfaces (που ορίζουν διάφορες συλλογές) και ένα σύνολο από κλάσεις που υλοποιούν ευρέως επαναχρησιμοποιούμενες **δομές δεδομένων** (data structures).
- Παρέχεται με την μορφή μιας βιβλιοθήκης.
- Τύποι συλλογών
  - **Ordered lists** (συλλογές ορισμένης σειράς αποθήκευσης και ανάκλησης)
  - **Dictionaries/Maps** (χρησιμοποιούν ένα lookup key για πρόσβαση στα στοιχεία της συλλογής)
  - **Sets** (αταξινόμητες συλλογές collections that can be iterated, contain each element at most once)

Interface Collection <E>

public interface Collection<E> extends Iterable<E>

- Είναι το root interface της ιεραρχίας συλλογών.
- Χρησιμοποιείται συνήθως για την διακίνηση και διαχείριση συλλογών όπου απαιτείται γενικότητα (maximum generality is desired).
- Κάποιες συλλογές επιτρέπουν όμοια στοιχεία, άλλες όχι.
- Κάποιες συλλογές έχουν τα στοιχεία τους σε σειρά (**ordered**), άλλες όχι. and
- Το JDK δεν δίνει κάποια άμεση υλοποίηση του interface Collection. Δίνει όμως υλοποιήσεις εξειδικεύσεων του όπως των **Set** και **List**.

boolean	add(E e)	Ensures that this collection contains the specified element (optional operation).
boolean	contains(Object o)	Returns true if this collection contains the specified element.
Iterator<E>	iterator()	Returns an iterator over the elements in this collection.

Interface – default and static methods

- Μέθοδοι που ορίζονται με τον προσδιοριστή **default** (java 8)
- Χρησιμοποιούνται από τα interfaces για να δώσουν default υλοποιήσεις μεθόδων.
- Μας επιτρέπουν να προσθέσουμε λειτουργικότητα στα interface της βιβλιοθήκης μας διασφαλίζοντας δυαδική συμβατότητα (***binary compatibility***) με προηγούμενες εκδόσεις των interfaces.
- Όταν κληρονομείς interface με default μέθοδο μπορείς:
  - Να μην αναφέρεις καθόλου την default μέθοδο, οπότε την κληρονομείς
  - Να δηλώσεις την default μέθοδο, οπότε την κάνεις abstract
  - Να ορίσεις ξανά την μέθοδο, οπότε την υπερκαλύπτεις (overrides)
- interface **static method**: όπως οι default μέθοδοι με τη διαφορά ότι οι κλάσεις που υλοποιούν το interface δεν μπορούν να τις υπερκαλύψουν.

Default Method Example 1

```
public interface Vehicle {
    void setUp();

    default void start() {
        System.out.println(" start method executed ");
    }
}

public class Car implements Vehicle {
    @Override
    public void setUp() {
        System.out.println(" setUp method executed ");
    }
}
```

```
public class CarVerhicleTest {
    static Car car;
    public static void main(String args[]){
        car = new Car();
        car.setUp();
        car.start();
    }
}
```

Default Method Example 2

```java
// A simple Java program to demonstrate multiple
// inheritance through default methods.
interface PI1
{
    // default method
    default void show()
    {
        System.out.println("Default PI1");
    }
}

interface PI2
{
    // Default method
    default void show()
    {
        System.out.println("Default PI2");
    }
}
```

```java
// Implementation class code
class TestClass implements PI1, PI2
{
    // Overriding default show method
    public void show()
    {
        // use super keyword to call the show
        // method of PI1 interface
        PI1.super.show();

        // use super keyword to call the show
        // method of PI2 interface
        PI2.super.show();
    }

    public static void main(String args[])
    {
        TestClass d = new TestClass();
        d.show();
    }
}
```

If we remove implementation of default method from "TestClass", we get compiler error.

Source: https://www.geeksforgeeks.org/java-and-multiple-inheritance/