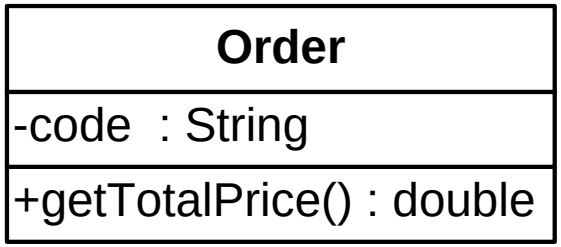
Φροντιστήριο JAVA

Οντοκεντρικός Προγραμματισμός (lect 2)

Ελένη Βογιατζάκη
evoyiatzaki@ceid.upatras.gr

Ορισμός κλάσης

- Κάθε κλάση κατά προτίμηση τοποθετείται σε ξεχωριστό αρχείο .java με όνομα το όνομα της κλάσης
- Ορατότητα ιδιοτήτων και μεθόδων
  - -: ιδιωτική ορατότητα (private)
  - +: δημόσια ορατότητα (public)




```
public class Order {

  private String code;

  public double getTotalPrice()
{
      //σώμα της μεθόδου
      //σχόλιο μιας γραμμής
      /* σχόλιο πολλών γραμμών
         με αυτό το σύμβολο */
}
```

Κατασκευαστές (Constructors)

- Κάθε κλάση διαθέτει μία «ειδική» μέθοδο που καλείται κατά την κατασκευή νέου αντικειμένου της κλάσης (κατασκευαστής – constructor)
- Αν δεν δηλωθεί κατασκευαστής, χρησιμοποιείται ο εξ' ορισμού κατασκευαστής που απλώς δημιουργεί το αντικείμενο χωρίς να αρχικοποιεί τις τιμές των ιδιοτήτων
- Ένας κατασκευαστής πρέπει υποχρεωτικά να έχει το όνομα της κλάσης
- Σε έναν κατασκευαστή με παραμέτρους μπορούμε να αποδώσουμε αρχικές τιμές σε ιδιότητες

Κατασκευαστές (Constructors)

κλάση

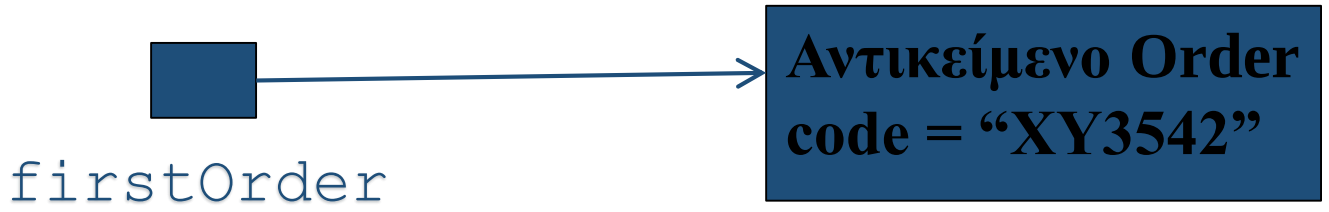
Order
- code : String
+ Order(text : String)

Constructor

```
public class Order {

   private String code;


   public Order(String text)  {

       code = text;

   }
}
```

Δημιουργία Αντικειμένων

- Δημιουργούμε ένα αντικείμενο κλάσης στη Java με χρήση της δεσμευμένης λέξης new και κλήση του κατασκευαστή της κλάσης
- Το νέο αντικείμενο το αναθέτουμε σε μία αναφορά (reference) του τύπου της κλάσης:
- **Order firstOrder; //δημιουργία αναφοράς προς αντικείμενο τύπου Order**
- **firstOrder = new Order("XY3542"); //ανάθεση τιμής στην αναφορά**
- Το firstOrder είναι μία αναφορά τύπου Order. Κατ' ουσία πρόκειται για έναν δείκτη (pointer) προς αντικείμενο τύπου Order



- Καταχρηστικά, αναφερόμαστε συνήθως στο ίδιο το αντικείμενο ως firstOrder

Δημιουργία αντικειμένων

- Στη Java, τα πάντα είναι αντικείμενα (και κατά συνέπεια τα χειριζόμαστε μέσω αναφορών σε αυτά)
  - String name = “Markos”;
    το name είναι αναφορά προς αντικείμενο τύπου String
  - int[] anArray = new int[10];
    το anArray είναι αναφορά προς αντικείμενο τύπου int[], δηλ. πίνακας ακεραίων

- … εκτός από 8 Στοιχειώδεις τύπους που δεν είναι αντικείμενα:
  - byte, short, int, long, float, double, boolean, char

Βασικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη	Τιμές
boolean	true/false	1 byte	true, false
char	Χαρακτήρας (Unicode)	2 bytes	Γράμματα, αριθμοί, σημεία στίξης και άλλα σύμβολα
byte	Ακέραιος	1 byte	-128 έως 127
short	Ακέραιος	2 bytes	-32.768 έως 32.767
int	Ακέραιος	4 bytes	-2.147.483.648 έως 2.147.483.647
long	Ακέραιος	8 bytes	-9,223,372,036,854,775,808 έως….
float	Πραγματικός	4 bytes	1,4E-45 έως $3,4^{E}$+38
double	Πραγματικός	8 bytes	4,9E-324 έως 1,7 E+308

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής συνδέεται με αυτό το χώρο στη μνήμη.

Παράδειγμα ορισμού κλάσης

```
class Employee {

    private String name;

    private double salary;


    Employee (String n, double s) {
        name = n;
        salary = s;
    }

    void pay () {
        System.out.println("Pay the employee named " +
                                name + " $" + salary);
    }
    public String getName() { return name; } // getter
}
```

Δημιουργία μίας «κατασκευής»

Κατασκευαστής αντικειμένου

Εισαγωγή τιμών

Μέθοδοι (Συναρτήσεις)

Παράδειγμα ορισμού κλάσης

```
class Employee {
    // Πεδία ή ιδιότητες κλάσεις
    private String name;
    private double salary;  // Σύμφωνα με τις μεθόδους που ορίζονται μετά, κάποιος εκτός κλάσης δε μπορεί να τη διαβάσει ούτε να την αλλάξει

    // Κατασκευαστής (constructor)
    Employee (String n, double s) {
        name = n;
        salary = s;
    }

    // Μέθοδοι
    void pay () {
        System.out.println("Pay the employee named " +
                                name + " $" + salary); }
    public String getName()
            { return name; } // getter
}
```

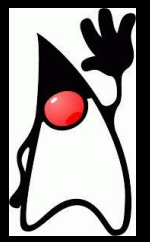
Δημιουργία αντικειμένων

- Όλα τα προγράμματα Java **πρέπει να περιλαμβάνουν μία μέθοδο main από την οποία ξεκινά η εκτέλεση** του προγράμματος
  - Στη main **μπορούμε να δημιουργήσουμε αντικείμενα των υπόλοιπων κλάσεων**
- **Η μέθοδος main μπορεί να βρίσκεται σε οποιαδήποτε κλάση. Συνήθως την τοποθετούμε μέσα σε μια κλάση με όνομα Main**

```
public class Main{

   public static void main(String[] args) {

      //Δημιουργία αντικειμένων

   }
}
```

HELLO WORLD!

Το έχουμε ξαναδεί!



Δομή ενός απλού Java προγράμματος

- Το όνομα του αρχείου που κρατάει το πρόγραμμα είναι X.java (όπου X το όνομα του προγράμματος)
  - Στο παράδειγμα μας ονομάζουμε το πρόγραμμα μας: HelloWorld.java
- Μέσα στο πρόγραμμα μας πρέπει να έχουμε μια κλάση με το όνομα X.
  - `class X` (`class HelloWorld` στο παράδειγμα μας)
- Η κλάση X θα πρέπει να περιέχει μια μέθοδο main η οποία είναι το σημείο εκκίνησης του προγράμματος μας
  - `public static void main(String[] args)`

File HelloWorld.java

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Το όνομα του .java αρχείου και το όνομα της κλάσης (που περιέχει την μέθοδο main) θα πρέπει να είναι πάντα τα ίδια!

Ορίζει την κλάση

Όνομα της κλάσης

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

```
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

Τα άγκιστρα { … } ορίζουν ένα **λογικό block** του κώδικα

- Αυτό μπορεί να είναι μία κλάση, μία συνάρτηση, ένα if statement
- **Οι μεταβλητές που ορίζουμε μέσα σε ένα λογικό block, έχουν εμβέλεια μέσα στο block**

Ορισμός της μεθόδου main

```
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

public, static: θα τα εξηγήσουμε σε επόμενο μάθημα

void: Η μέθοδος δεν επιστρέφει τίποτα. Η main δεν επιστρέφει κάτι οπότε είναι πάντα void

main: σηματοδοτεί το σημείο εκκίνησης του προγράμματος

Ορισμός της μεθόδου main

```
/**
* A class that prints a message "hello world"
*/
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

Ορίσματα της μεθόδου

- Ένας πίνακας από Strings που αντιστοιχούν στις παραμέτρους με τις οποίες τρέχουμε το πρόγραμμα.
- **String**: κλάση βιβλιοθήκης της Java που χειρίζεται τα αλφαριθμητικά

```
/**
* <h1>Hello, World!</h1>
* The HelloWorld program implements an application that
* simply displays "Hello World!" to the standard output.
* <p>
* Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
```

```
{
    public static void main(String args[])
    {
            // σχόλιο 1 γραμμής
            System.out.println("Hello world!");
        }
}
```

Κάθε εντολή στη Java πρέπει να τερματίζει με το ;

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Αντικείμενο `System.out`

Μέθοδος `println`:
Τυπώνει το String που δίνεται ως όρισμα και αλλάζει γραμμή

http://javapapers.com/core-java/system-out-println/

Παράδειγμα

- Φτιάξτε ένα πρόγραμμα που τυπώνει το αποτέλεσμα της διαίρεσης δύο ακεραίων.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ορισμός μεταβλητών

- Η Java είναι strongly typed γλώσσα: κάθε μεταβλητή θα πρέπει να έχει ένα τύπο
- Οι τύποι int και double είναι βασικοί τύποι (primitive types)
- Εκτός από τους βασικούς τύπους, όλοι οι άλλοι τύποι είναι κλάσεις

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division  = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ανάθεση: αποτίμηση της τιμής της έκφρασης στο δεξιό μέλος του "=" και μετά ανάθεση της τιμής στην μεταβλητή στο αριστερό μέλος

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
//Θα εμφανίσει 3.2
```

Μετατροπή τύπου:
(double)denominator μετατρέπει την τιμή της μεταβλητής denominator σε double.
Αν δεν γίνει η μετατροπή, η διαίρεση μεταξύ ακεραίων μας δίνει πάντα ακέραιο.

```
division = enumerator/denominator;
System.out.println("Result = " + division);
//Θα εμφανίσει 3
```

Αναθέσεις

- Στην ανάθεση κατά κανόνα, η τιμή του δεξιού μέρους θα πρέπει να είναι ίδιου τύπου με την μεταβλητή του αριστερού μέρους.
- Υπάρχουν εξαιρέσεις όταν υπάρχει συμβατότητα μεταξύ τύπων

- byte → short → int → long → float → double
  - Μια τιμή τύπου T μπορούμε να την αναθέσουμε σε μια μεταβλητή τύπου που εμφανίζεται δεξιά του T

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ο τελεστής "+" μεταξύ αντικείμενων της κλάσης String συνενώνει (concatenates) τα δύο String.
Μεταξύ ενός String και ενός βασικού τύπου, ο βασικός τύπος μετατρέπεται σε String και γίνεται η συνένωση

Αλφαριθμητικά (strings)

- Η κλάση **String** είναι προκαθορισμένη κλάση της Java που μας επιτρέπει να χειριζόμαστε αλφαριθμητικά.
- Ο τελεστής "+" μας επιτρέπει την συνένωση
- Υπάρχουν πολλές χρήσιμες μέθοδοι της κλάσης String.
  - length(): μήκος του String
  - equals(String x): ελέγχει για ισότητα του αντικειμένου που κάλεσε την μέθοδο και του ορίσματος x.
  - indexOf("x"): Επιστρέφει τη θέση της πρώτης εμφάνισης του string x
  - trim(): αφαιρεί κενά στην αρχή και το τέλος του string.
  - κ.α.

```
public class MyClass {
  public static void main(String[] args) {
    String txt = " ABCDEFGHI ";
    System.out.println("The length of the txt string is: " + txt.length());
    System.out.println("the position of the character is "+txt.indexOf("A"));
    System.out.println("the position of the character is "+txt.indexOf("W"));
    System.out.println(txt.trim( ));

  }
}
```

```
The length of the txt string is: 11
the position of the character is 1
the position of the character is -1
ABCDEFGHI
```

Αν ειχαμε δευτερο string
String myStr = “abc”
System.out.println(myStr.equals(txt)); // θα επεστρεφε false

String

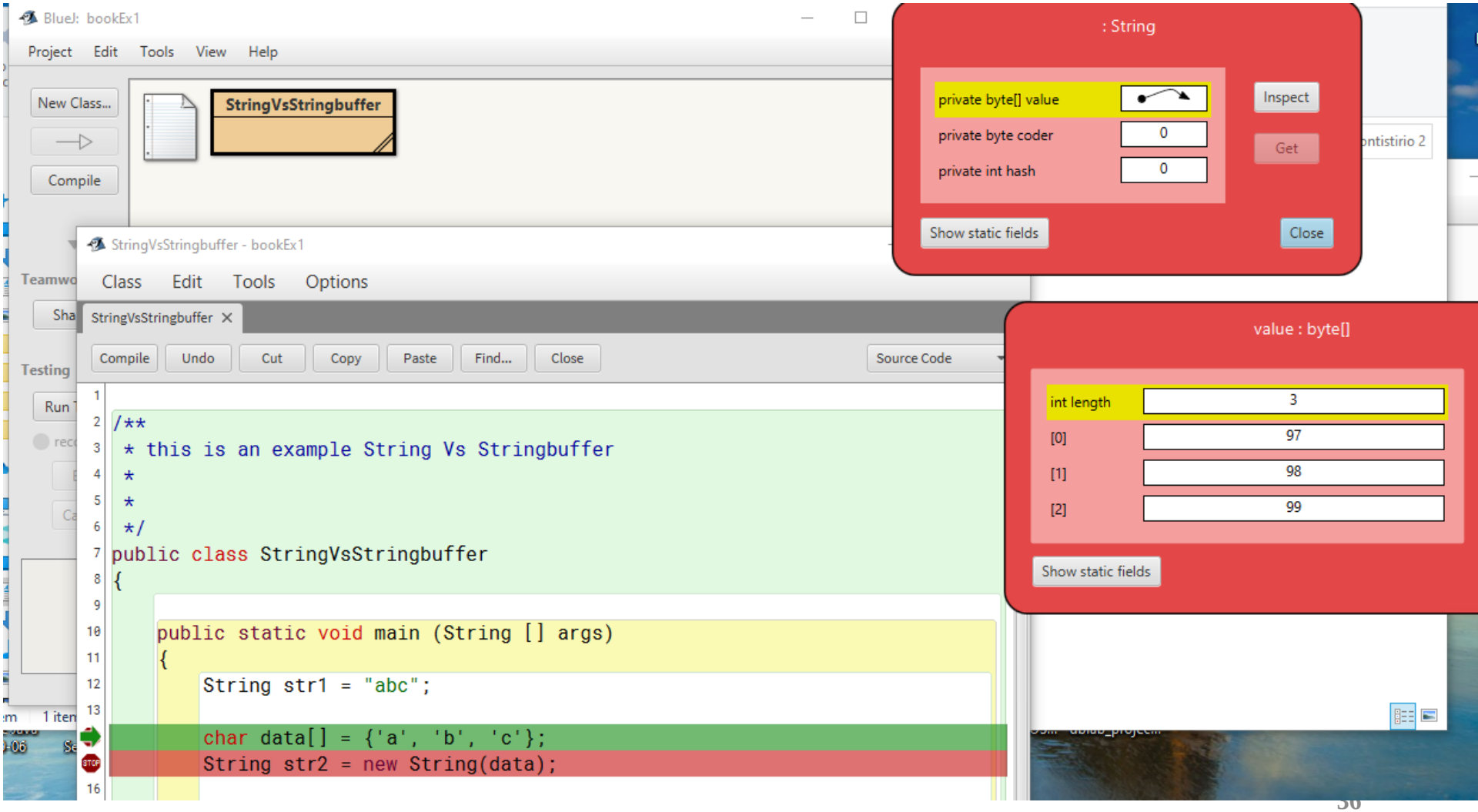
- The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.
- Strings are constant; their values cannot be changed after they are created.
- Because String objects are immutable they can be shared. For example:

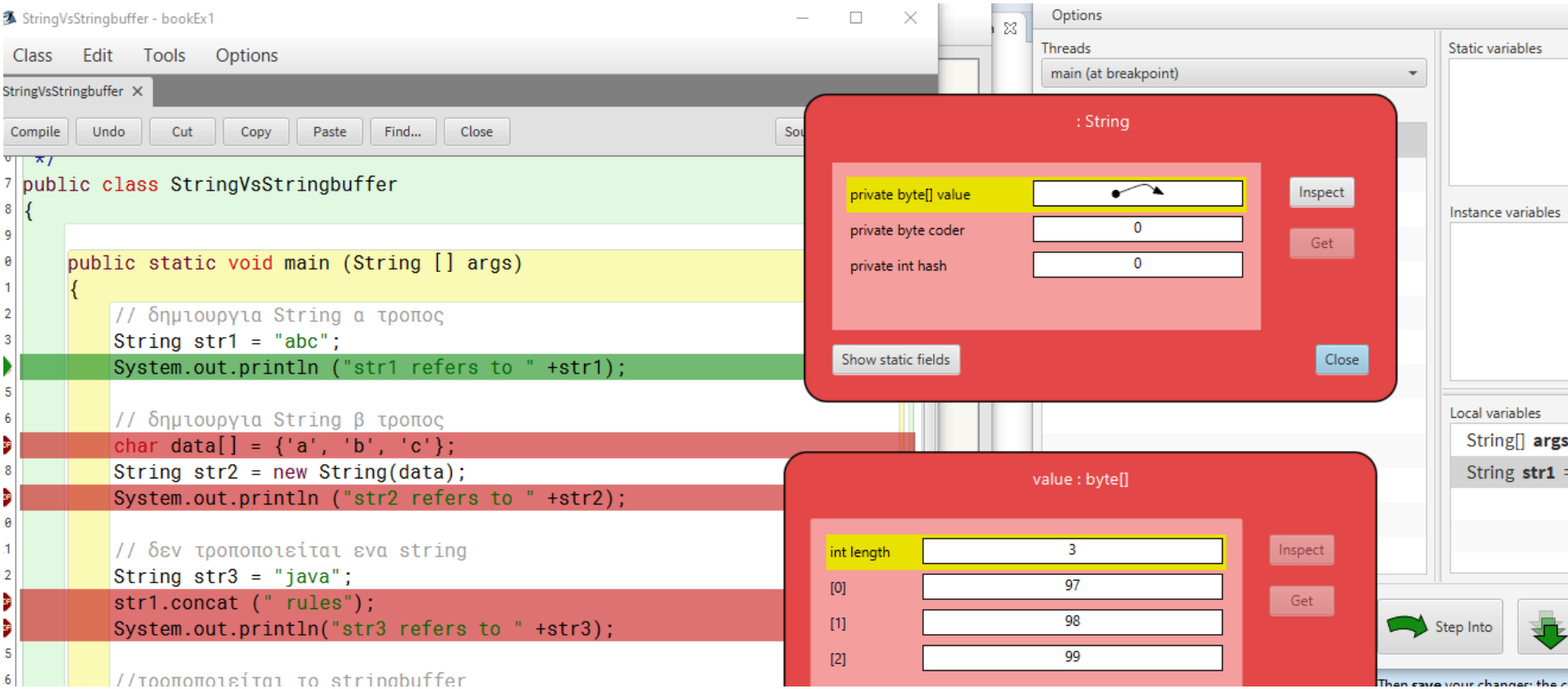
```
String str = "abc";
```

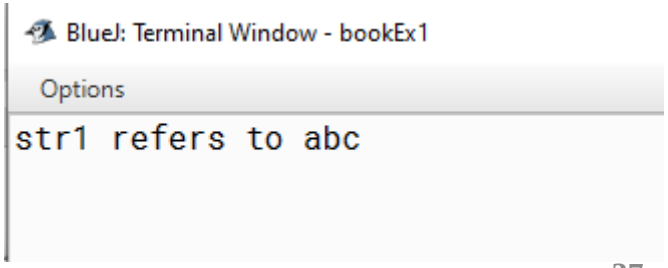
is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

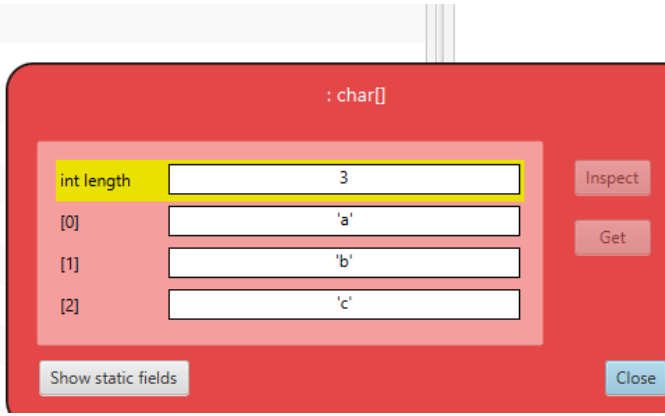
Παράδειγμα

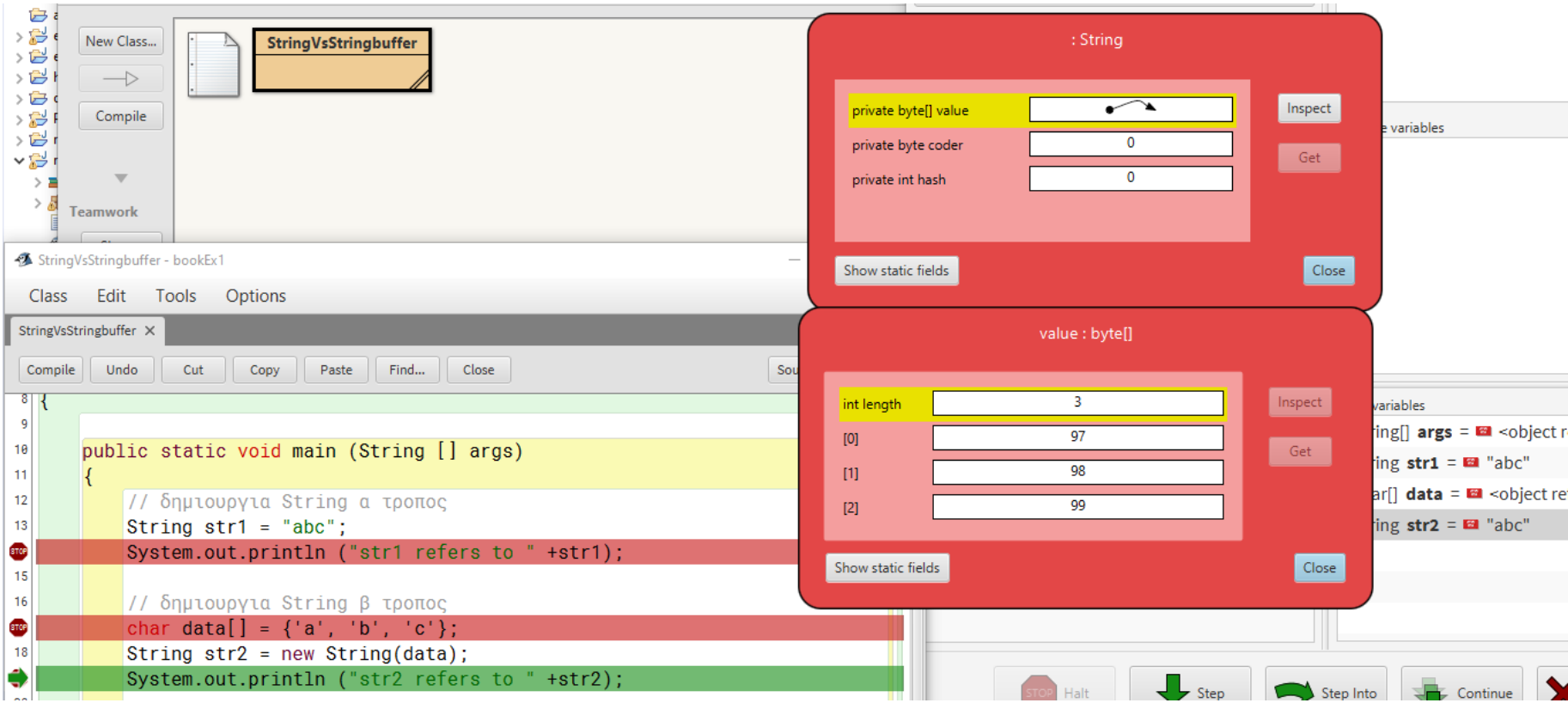

BlueJ: bookEx1
Project Edit Tools View Help
New Class...
Compile
StringVsStringbuffer
: String
private byte[] value
private byte coder
0
private int hash
0
Inspect
Get
Show static fields
Close
StringVsStringbuffer - bookEx1
Class Edit Tools Options
StringVsStringbuffer
Compile Undo Cut Copy Paste Find... Close
Source Code
/**
* this is an example String Vs Stringbuffer
*
*
*/
public class StringVsStringbuffer
{
public static void main (String [] args)
{
String str1 = "abc";
char data[] = {'a', 'b', 'c'};
String str2 = new String(data);
value : byte[]
int length
3
[0]
97
[1]
98
[2]
99
Show static fields

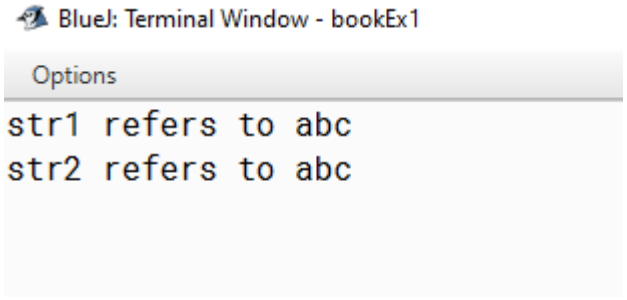

StringVsStringbuffer - bookEx1
Class Edit Tools Options
StringVsStringbuffer
Compile Undo Cut Copy Paste Find... Close
*/
public class StringVsStringbuffer
{
public static void main (String [] args)
{
// δημιουργια String α τροπος
String str1 = "abc";
System.out.println ("str1 refers to " +str1);
// δημιουργια String β τροπος
char data[] = {'a', 'b', 'c'};
String str2 = new String(data);
System.out.println ("str2 refers to " +str2);
// δεν τροποποιειται ενα string
String str3 = "java";
str1.concat (" rules");
System.out.println("str3 refers to " +str3);
//τροποποιείται το stringbuffer
Options
Threads
main (at breakpoint)
Static variables
Instance variables
Local variables
String[] args
String str1
: String
private byte[] value
private byte coder
0
private int hash
0
Inspect
Get
Show static fields
Close
value : byte[]
int length
3
[0]
97
[1]
98
[2]
99
Inspect
Get
Step Into


BlueJ: Terminal Window - bookEx1
Options
str1 refers to abc

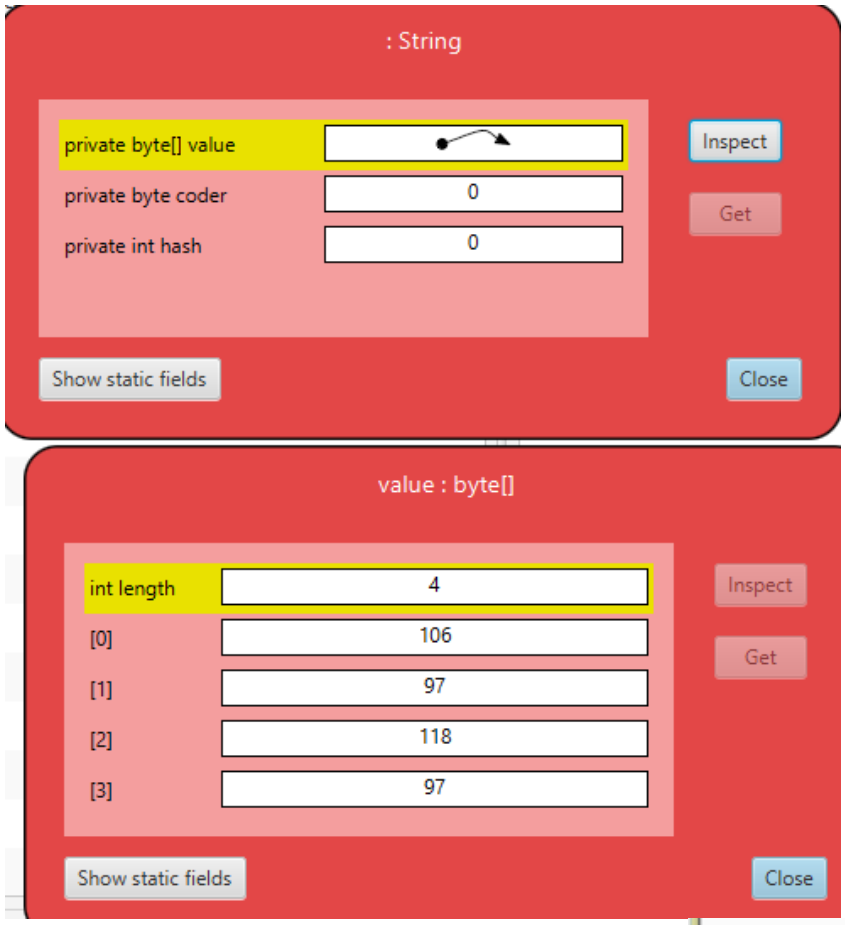
```
10  public static void main (String [] args)
11  {
12      // δημιουργια String α τροπος
13      String str1 = "abc";
        System.out.println ("str1 refers to " +str1);
15
16      // δημιουργια String β τροπος
        char data[] = {'a', 'b', 'c'};
        String str2 = new String(data);
        System.out.println ("str2 refers to " +str2);
20
21      // δεν τροποποιείται ενα string
22      String str3 = "java";
```




New Class...
Compile
Teamwork
StringVsStringbuffer
StringVsStringbuffer - bookEx1
Class
Edit
Tools
Options
StringVsStringbuffer
Compile
Undo
Cut
Copy
Paste
Find...
Close
{
public static void main (String [] args)
{
// δημιουργια String α τροπος
String str1 = "abc";
System.out.println ("str1 refers to " +str1);
// δημιουργια String β τροπος
char data[] = {'a', 'b', 'c'};
String str2 = new String(data);
System.out.println ("str2 refers to " +str2);
: String
private byte[] value
private byte coder
0
private int hash
0
Inspect
Get
Show static fields
Close
value : byte[]
int length
3
[0]
97
[1]
98
[2]
99
Inspect
Get
Show static fields
Close
variables
ring[] args = <object
ring str1 = "abc"
ar[] data = <object re
ring str2 = "abc"
Halt
Step
Step Into
Continue


BlueJ: Terminal Window - bookEx1
Options
str1 refers to abc
str2 refers to abc

```
String str3 = "java";
str3.concat (" rules");
System.out.println("str3 refers to " +str3);
```




```
str3 refers to java
```

Ρεύματα εισόδου/εξόδου

- Τι είναι ένα ρεύμα (stream)? Μια αφαίρεση που αναπαριστά μια πηγή (για την είσοδο), ή ένα προορισμό (για την έξοδο) χαρακτήρων
  - Αυτό μπορεί να είναι ένα αρχείο, το πληκτρολόγιο, η οθόνη.
  - Όταν δημιουργούμε το ρεύμα το συνδέουμε με την ανάλογη πηγή, ή προορισμό.

Είσοδος & Έξοδος

- Τα βασικά ρεύματα εισόδου/εξόδου είναι έτοιμα αντικείμενα τα οποία ορίζονται σαν πεδία (στατικά) της κλάσης System
  - System.out
  - System.in
  - System.err
- Μέσω αυτών και άλλων βοηθητικών αντικειμένων γίνεται η είσοδος και έξοδος δεδομένων ενός προγράμματος.
- Μια εντολή εισόδου/εξόδου έχει αποτέλεσμα το λειτουργικό να πάρει ή να στείλει χαρακτήρες από/προς την αντίστοιχη πηγή/προορισμό.

Είσοδος

- Χρησιμοποιούμε την κλάση Scanner της Java
  - import java.util.Scanner;
- Αρχικοποιείται με το ρεύμα εισόδου:
  - Scanner input = new Scanner(System.in);
- Μπορούμε να καλέσουμε μεθόδους για να διαβάσουμε κάτι από την είσοδο
  - nextLine(): διαβάζει μέχρι να βρει τον χαρακτήρα '\n'
  - next(): διαβάζει το επόμενο String
  - nextInt(): διαβάζει τον επόμενο int
  - nextDouble(): διαβάζει τον επόμενο double.

Παράδειγμα

```
import java.util.Scanner;


public class TestIO
{
      public static void main(String args[])
      {
            System.out.print("Say Something: ");
            Scanner input = new Scanner(System.in);
            String line = input.nextLine();
            System.out.println("You said: " + line);
      }
}
```

new: δημιουργεί ένα αντικείμενο τύπου Scanner (μία μεταβλητή) με το οποίο μπορούμε πλέον να διαβάζουμε από την είσοδο

Έξοδος

- Μπορούμε να καλέσουμε τις μεθόδους του αντικειμένου System.out:
  - println(String s): για να τυπώσουμε ένα αλφαριθμητικό s και τον χαρακτήρα '\n' (αλλαγή γραμμής)
  - print(String s): τυπώνει το s αλλά δεν αλλάζει γραμμή
  - printf: Formatted output
    - printf("%d",myInt); // τυπώνει ένα ακέραιο
    - printf("%f",myDouble); // τυπώνει ένα πραγματικό
    - printf("%.2f",myDouble); // τυπώνει ένα πραγματικό με δύο δεκαδικά

Είσοδος από το πληκτρολόγιο

- Χρησιμοποιούμε την κλάση Scanner της Java
  - import java.util.Scanner;
- Αρχικοποιείται με το ρεύμα εισόδου:
  - Scanner input = new Scanner(System.in);
- Μπορούμε να καλέσουμε μεθόδους της Scanner για να διαβάσουμε κάτι από την είσοδο
  - nextLine(): διαβάζει μέχρι να βρει τον χαρακτήρα '\n'
  - next(): διαβάζει το επόμενο String
  - nextInt(): διαβάζει τον επόμενο int
  - nextDouble(): διαβάζει τον επόμενο double.

Παράδειγμα

```
import java.util.Scanner; // Import the Scanner class


public class TestIO
{
      public static void main(String args[])
      {
            Scanner input = new Scanner(System.in);
            String line = input.nextLine();
            System.out.println(line);
      }
}
```

καλημερα
καλημερα

new: δημιουργεί ένα αντικείμενο τύπου `Scanner` με το οποίο μπορούμε πλέον να διαβάζουμε από την είσοδο

Division.java με είσοδο από το χρήστη

```
import java.util.Scanner; // Import the Scanner class


public class Division {
  public static void main(String args[])   {
// Δημιουργεί το Scanner για να πάρει είσοδο από την κονσόλα
        Scanner input = new Scanner(System.in);
        int enumerator;
        int denominator;
        double division;
        System.out.println("Enter first integer: ");
        enumerator = input.nextInt();
        System.out.println(" Enter second integer: ");
        denominator = input.nextInt();


        division = enumerator/(double)denominator;
        System.out.println(" Result = " + division);
  }
}
```

```
Enter first integer:
3
 Enter second integer:
2
Result = 1.5
```

Μέχρι στιγμής τίποτα αντικειμενοστραφές!

- Δεν κατασκευάσαμε κλάσεις και αντικείμενα
- Αν και χρησιμοποιήσαμε κάποιες κλάσεις βιβλιοθήκης της Java (String, Scanner) και κάποιες ‘έτοιμες’ μεθόδους
- Μελετήστε ένα παράδειγμα κώδικα σε δύο εκδοχές
  - μια μη αντικειμενοστρεφή
  - μια αντικειμενοστρεφή

Παράδειγμα

- Ζητούμενη λειτουργικότητα
  - Δίνουμε σαν είσοδο χαρακτηριστικά προϊόντων
    - Όνομα
    - Τιμή
    - Σκορ (σαν αξιολόγηση)
  - Αυτό γίνεται μέχρι να εισάγουμε ένα συγκεκριμένο πλήθος προϊόντων (έως να δοθεί ένδειξη τέλους εισαγωγής)
  - Το πρόγραμμα υπολογίζει από όλα τα προϊόντα το καλύτερο (αυτό με τον μεγαλύτερο σκορ/τιμη)
  - Εμφανίζει τα στοιχεία του προϊόντος αυτού

Λύση χωρίς Αντικειμενο-στρέφεια

```
import java.util.Scanner;

public class Main {

        public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        String best_name = "";
        double best_price = 1;
        int best_score = 0;

        boolean more = true;

        while(more) {

                String next_name;
                double next_price;
                int next_score;

                System.out.println("Please enter the product name: ");
                next_name = in.nextLine();
                System.out.println("Please enter the product price: ");
                next_price = in.nextDouble();
                System.out.println("Please enter the product score: ");
                next_score = in.nextInt();

                if(next_score/next_price > best_score/best_price) {
                        best_name = next_name;
                        best_price = next_price;
                        best_score = next_score;
                }

                System.out.println("More data ? (1=YES, 2=NO)");
                int answer = in.nextInt();
                if(answer != 1)
                        more = false;
                in.nextLine();
                }

System.out.println("The best product is: " + best_name);
System.out.println("The best price is: " + best_price);
System.out.println("The best score is: " + best_score);
```

Αρχικοποίηση

Επαναληπτική διαδικασία

Εισαγωγή δεδομένων

Έλεγχος και αλλαγή της best

Εμφάνιση αποτελεσμάτων

Αντικειμενο-στρέφεια Κλάση Προϊόν

```
import java.util.Scanner;

public class Product {
   private String name;
   private double price;
   private int score;
public Product (){
   name = "";
   price = 1;
   score = 0;
}
public void read (){
  Scanner in = new Scanner(System.in);
   System.out.println("Please enter the product name: ");
                name = in.nextLine();
                System.out.println("Please enter the product price: ");
                price = in.nextDouble();
                System.out.println("Please enter the product score: ");
                score = in.nextInt();
}
public boolean isBetterThan(Product other) {
   if ((score/price)>(other.score/other.price))
   return true;
   else return false;
}

public void print () {
   System.out.println("Product Name: " + name);
                System.out.println("Price: " + price);
                System.out.println("Score: " + score);
 }
}
```

Αρχικοποίηση μέσα στον constructor

Εισαγωγή δεδομένων

Μέθοδοι

Έλεγχος και αλλαγή της best

Εμφάνιση αποτελεσμάτων

Αντικειμενο-στρέφεια main

```
import java.util.Scanner;
public class Main {

   public static void main(String[] args) {

                    Scanner in = new Scanner(System.in);

Αρχικοποίηση        Product best = new Product();          Δημιουργία instance

                    boolean more = true;
                    while(more) {
                                                  Κλήση μεθόδου για
                        Product current = new Product(); Εισαγωγή δεδομένων
Επαναληπτική        current.read();
διαδικασία
                        if (current.isBetterThan(best))   Κλήση μεθόδου για
                                best = current;           Έλεγχο και αλλαγή της
                                                          best
                        System.out.println("More data ? (1=YES, 2=NO)");
                        int answer = in.nextInt();
                        if(answer != 1)
                                more = false;
                        in.nextLine();

                    }

                    best.print();
                                  Κλήση μεθόδου για
                                  Εμφάνιση αποτελεσμάτων
        }

}
```

Δημιουργία instance

RpnCalculator Δραστηριότητα2

RpnCalculator

- Διαβάζει μια έκφραση με ακέραιους αριθμούς και τελεστές
- Ελέγχει ένα ένα χαρακτήρα
  - Αν είναι αριθμός **τον βάζει στο stack**
  - Αν είναι τελεστής,
    - Ελέγχει ποιος είναι για να εκτελέσει την αντίστοιχη πράξη
      - **παίρνει τους δυο «πάνω» αριθμούς από το stack**
      - **Κανει την πράξη**
      - **Το αποτέλεσμα το βάζει στο stack**
  - **Αν είναι = εμφανίζει το αποτέλεσμα**

- getExpression() Διαβάζει μια έκφραση με ακέραιους αριθμούς και τελεστές
- Ελέγχει ένα ένα χαρακτήρα getOp() (**SWITCH**)
  - Αν είναι αριθμός **τον βάζει στο stack push()**
  - Αν είναι τελεστής,
    - Ελέγχει ποιος είναι για να εκτελέσει την αντίστοιχη πράξη (**add(),sub()**)
      - **παίρνει τους δυο «πάνω» αριθμούς από το stack pop()**
      - **Κάνει την πράξη**
      - **Το αποτέλεσμα το βάζει στο stack push()**
  - **Αν είναι = εμφανίζει το αποτέλεσμα displayResult()**

```
public class RpnCalc
{   ……….
   public static void main(String[] args) {
      stack = new int[20];
      sp = 0;
      expresion = getExpression();
      sc = new Scanner(expresion);
      sc.useDelimiter(" ");
      while(sc.hasNext()){
         inputType = getOp();
               SWITCH (add(), sub(), displayResult()
   static String getExpression() {
      sc = new Scanner(System.in);
      System.out.print("Expression to evaluate:");
      expresion = sc.nextLine();
      sc.close();
      return expresion;
……..
   }
```

```
private static void displayresult() {
}
```

```
private static void add() {
}
private static void sub() {
}
```

```
private static int pop() {
}
private static void push(int integer) {
```

```
static int getOp() {
    input = sc.next();
    if(Character.isDigit(input.charAt(0)))
       return NUMBER;
    else
       return input.charAt(0);
    }
}
```

```
public class RpnCalc
{   ……
   public static void main(String[] args) {
      stack = new int[20];
      sp = 0;
      expresion = getExpression();
      sc = new Scanner(expresion);
      sc.useDelimiter(" ");
      while (sc.hasNext()){
         inputType = getOp();
               SWITCH (add(), sub(), displayResult()

static String getExpression() {
      sc = new Scanner(System.in);
      System.out.print("Expression to evaluate:");
      expresion = sc.nextLine();
      sc.close();
      return expresion;
……
   }
```

```
private static void displayresult() {
   }
```

```
private static void add() {
}
private static void sub() {
}
```

```
private static int pop() {
}
private static void push(int integer) {
```

```
static int getOp() {
    input = sc.next();
    if(Character.isDigit(input.charAt(0)))
       return NUMBER;
    else
       return input.charAt(0);
    }
}
```

```
{ ……
   public static void main(String[] args) {
      stack = new int[20];
      sp = 0;
      expresion = getExpression();
      sc = new Scanner(expresion);
      sc.useDelimiter(" ");
      while(sc.hasNext()){
         inputType = getOp();
switch(inputType) {
            case NUMBER:
               push(Integer.parseInt(input));
               break;
            case '+':
               add();
               break;
            case '-':
               sub();
               break;
            case '=':
               displayresult();
```

```
private static void displayresult() {

   }
   private static void add() {

   }
   private static void sub() {



   }
……..
```

```
   private static int pop() {


   }

   private static void push(int integer) {



   }
```

```
   static int getOp() {
      input = sc.next();
      if(Character.isDigit(input.charAt(0)))
         return NUMBER;
      else
         return input.charAt(0);
      }
}
```

```
import java.util.Scanner;


public class RpnCalc

{

    static final int NUMBER=1;

    static String expresion;

    static Scanner sc;

    static String input;

    static int inputType;

    static int [] stack;

    static int sp;


    public static void main(String[] args) {

        // TODO Auto-generated method stub

        System.out.println("Reverse Polish Notation
Calculator");

        stack = new int[20];

        sp = 0;

        expresion = getExpression();

        sc = new Scanner(expresion);

        sc.useDelimiter(" ");

        while(sc.hasNext()){

            inputType = getOp();
```

```
switch(inputType) {
                case NUMBER:
                    push(Integer.parseInt(input));
                    break;
                case '+':
                    add();
                    break;
                case '-':
                    sub();
                    break;
                case '=':
                    displayresult();
            }
        }
        sc.close();
    }
```

```
 static String getExpression() {
        sc = new Scanner(System.in);
        System.out.print("Expression to
evaluate:");
        expresion = sc.nextLine();
        sc.close();
        return expresion;
    }
 private static void displayresult() {
        System.out.println("result=" +pop());
    }

    private static void add() {
        push(pop()+pop());
    }

    private static void sub() {
        int n1;
        n1=pop();
        push(pop()-n1);
    }
    private static int pop() {
        // TODO Auto-generated method stub
        return stack[--sp];
    }

    private static void push(int integer) {
        // TODO Auto-generated method stub
        stack[sp] = integer;
        sp++;
    }

    static int getOp() {
        input = sc.next();
        if(Character.isDigit(input.charAt(0)))
            return NUMBER;
        else
            return input.charAt(0);
        }
}
```