
Καφέζας Γιώργος
Α.Μ. 4465



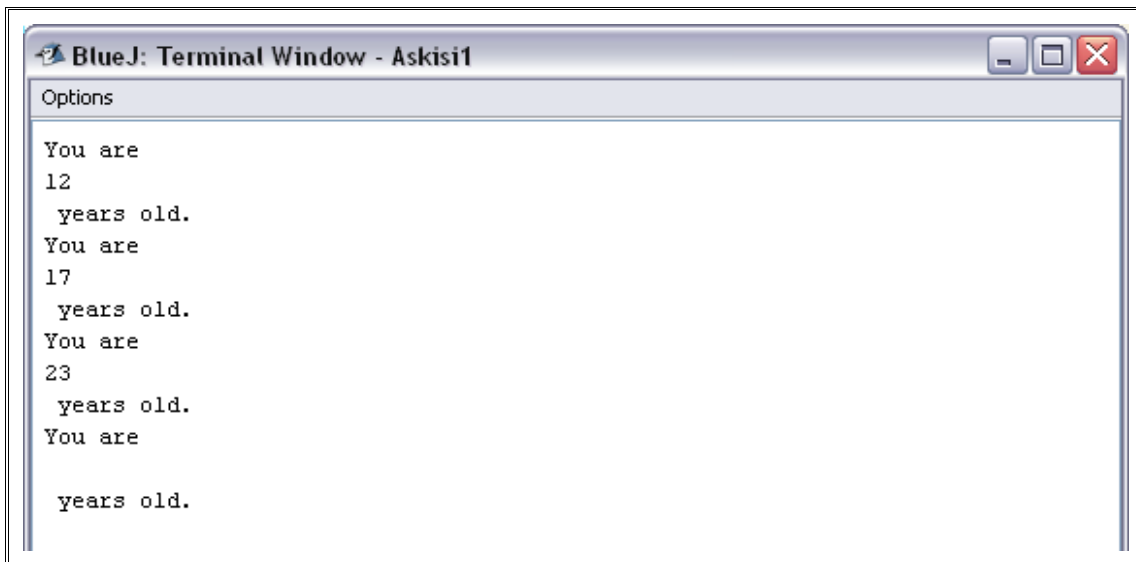
Εργαστήριο Οντοκεντρικού Προγραμματισμού I (JAVA)

Ακαδημαϊκό Έτος 2011-2012

1^ο Σετ Ασκήσεων

1^η Εργαστηριακή Άσκηση

3) Τρέχοντας το πρόγραμμα σύμφωνα με τις οδηγίες στο φυλλάδιο των εργαστηριακών ασκήσεων και δίνοντας ως ορίσματα για αρχή το «12» και στη συνέχεια το «17», το «23» και τέλος τίποτα, πήραμε τα παρακάτω αποτελέσματα, όπως εμφανίζονται συνολικά μετά από όλες τις εκτελέσεις του προγράμματος:

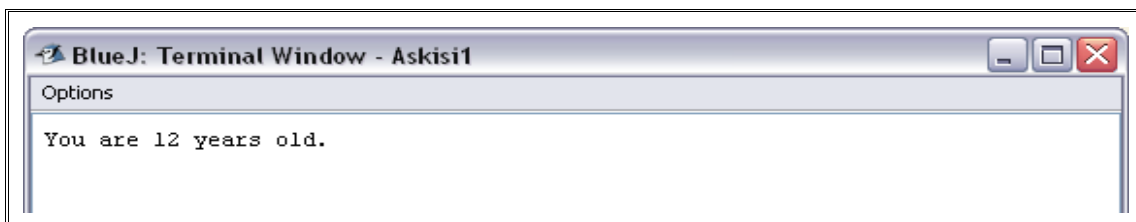


```
Options
You are
12
  years old.
You are
17
  years old.
You are
23
  years old.
You are

  years old.
```

Παρατηρούμε, λοιπόν, ότι όταν δώσουμε κάποιο όρισμα, π.χ. «12», τότε εκτυπώνεται το μήνυμα “You are 12 years old.”, ενώ όταν δε δώσουμε κάποιο όρισμα, τυπώνεται πάλι ένα μήνυμα, χωρίς όμως τον αριθμό που θα έπρεπε να αναφέρεται στην ηλικία, δηλαδή “You are __ years old.”.

4) Αλλάζοντας τον κώδικα του προγράμματος με το δοθέντα κομμάτι, ουσιαστικά αλλάξαμε την εντολή `System.out.println()` σε `System.out.print()`. Η διαφορά μεταξύ τους έγκειται στο ότι η πρώτη μόλις εκτυπώσει ό,τι είναι να εκτυπώσει εισάγει μια αλλαγή γραμμής, ενώ η δεύτερη όχι.



```
Options
You are 12 years old.
```

Ξαναδίνοντας ως όρισμα το «12» πήραμε το παραπάνω αποτέλεσμα, που φανερώνει ουσιαστικά την προηγούμενη παρατήρηση ως προς τη διαφορά μεταξύ των δύο εντολών.

5) Το κομμάτι κώδικα που δίνεται από το ερώτημα της εκφώνησης πρέπει να το τοποθετήσουμε ως εξής:

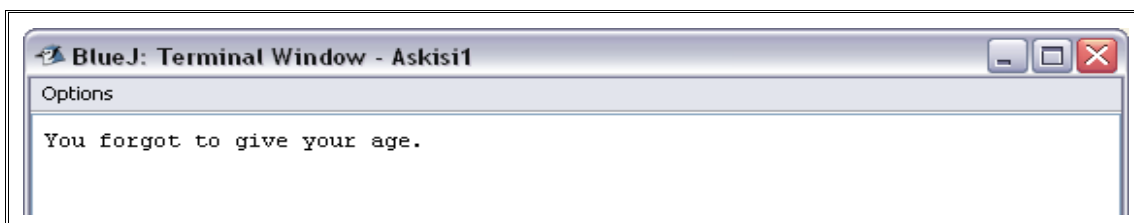
```
public class Age{
    public static void main(String args[]){
        if (args.length == 1){
            System.out.print("You are ");
        }
    }
}
```

```

        System.out.print(args[0]);
        System.out.println(" years old.");
    }else
        System.out.println("You forgot to give your age.");
    }
}

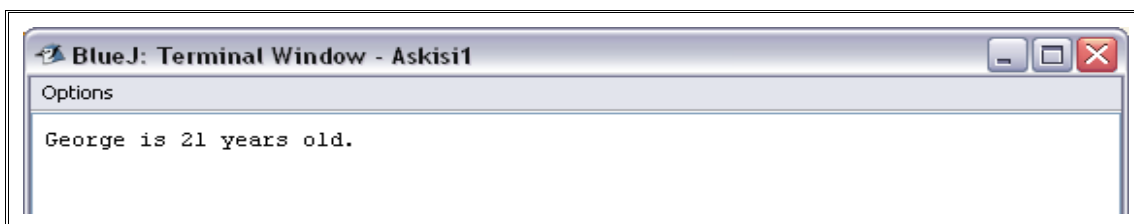
```

Έπειτα από την μετάφραση και το τρέξιμο του προγράμματος χωρίς να δώσουμε όρισμα, είχαμε το εξής:

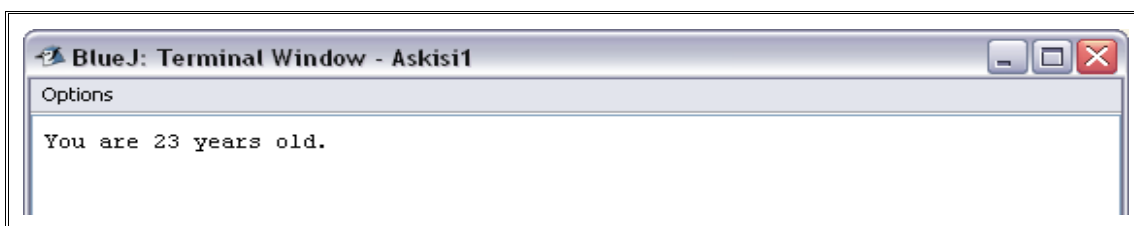


Οπότε βλέπουμε πως η τοποθέτηση του κομματιού του κώδικα στο σημείο που φαίνεται στον άνω κώδικα λειτουργεί κανονικά στην περίπτωση που δε δοθεί κάτι ως όρισμα.

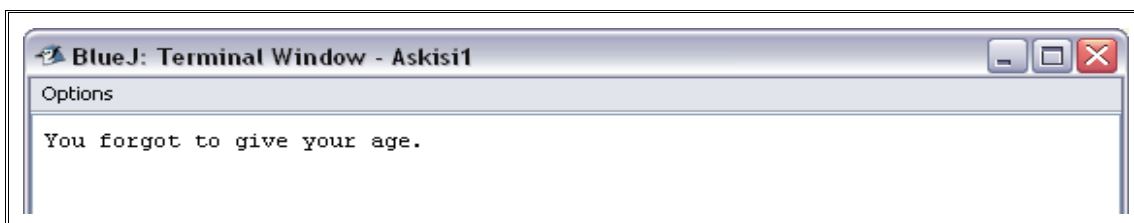
6) Κάνοντας τις προτεινόμενες αλλαγές, ξαναμεταφράζοντας και τρέχοντας το πρόγραμμα εκ νέου, δίνοντας ως ορίσματα την ηλικία «21» και το όνομα «George», πήραμε το παρακάτω αποτέλεσμα:



Δίνοντας όρισμα μόνο τον αριθμό «23» έχουμε πάλι ορθό αποτέλεσμα:



Τέλος, όταν δε δίνουμε κανένα όρισμα μας εμφανίζεται το αντίστοιχο μήνυμα:



7) Θεωρώντας ότι η αλλαγή θα αφορά την περίπτωση όπου δίνουμε δύο ορίσματα, τότε ο κώδικας θα μπορούσε να μετατραπεί ως κάτωθι, αλλάζοντας το `args[0]` με το `args[1]`:

```

System.out.print(args[1]);

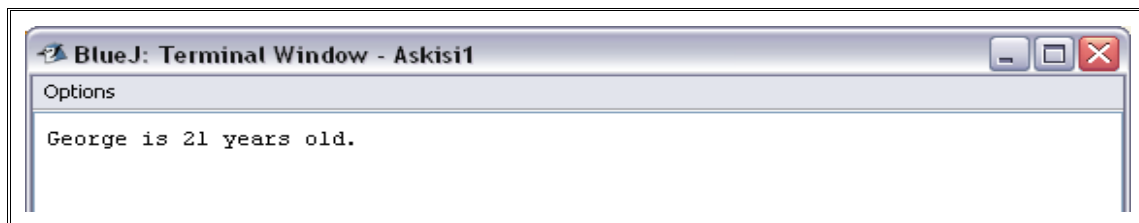
```

```
System.out.print(" is ");  
System.out.print(args[1]);  
System.out.println(" years old.");
```

Έτσι, δίνουμε πλέον και με την ανάλογη σειρά τα ορίσματα:



Και παίρνουμε τα αντίστοιχα αποτελέσματα:

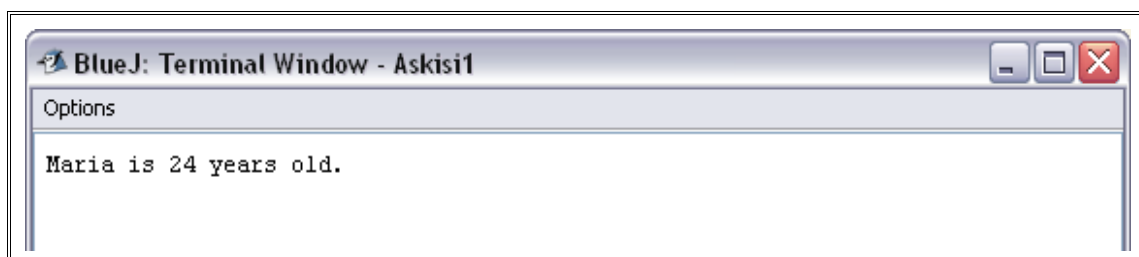


Συνεπώς το πρόγραμμα συνεχίζει να δουλεύει ορθά μετά την αλλαγή στον κώδικα.

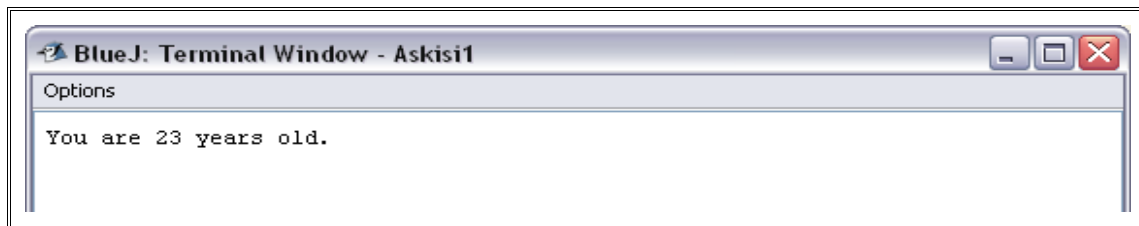
8) Αντικαθιστώντας και όσες από τις υπόλοιπες εντολές εκτύπωσης γινόταν σύμφωνα με τις υποδείξεις της εκφώνησης, προκύπτει το ακόλουθο πρόγραμμα:

```
public class Age{  
    public static void main(String args[]){  
        if (args.length == 1){  
            System.out.print("You are " + args[0] + " years old.");  
        }else if (args.length == 0){  
            System.out.println("You forgot to give your age.");  
        }else{  
            System.out.print(args[0] + " is " + args[1] + " years old.");  
        }  
    }  
}
```

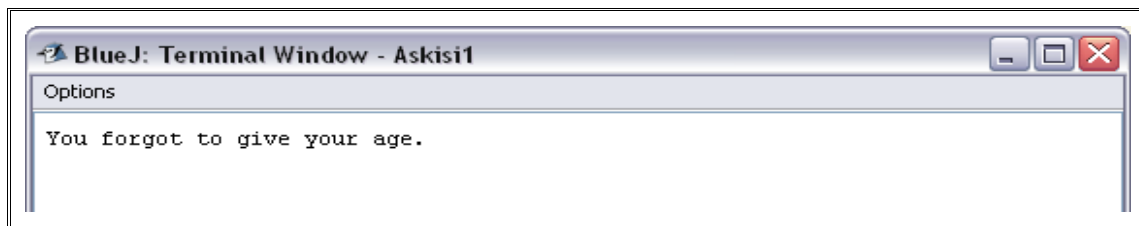
Δίνοντας τα ορίσματα «Maria» και «24» έχουμε το παρακάτω αποτέλεσμα:



Αντίστοιχα, δίνοντας μόνο ένα όρισμα, το «23», έχουμε το ακόλουθο αποτέλεσμα:



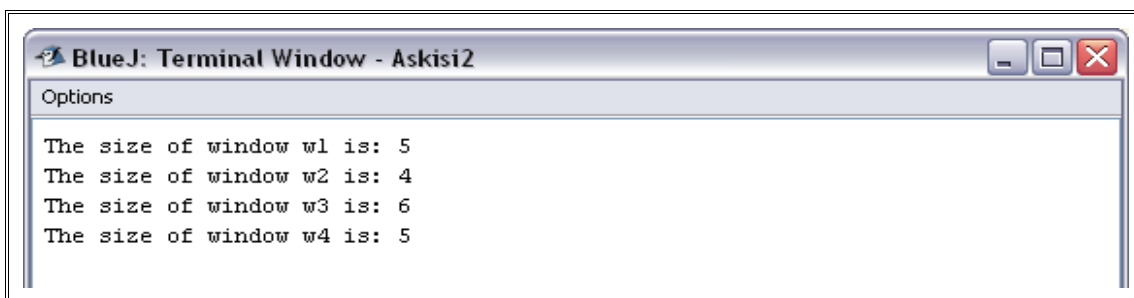
Και τέλος, αφήνοντας να τρέξει το πρόγραμμα χωρίς να του δώσουμε όρισμα έχουμε το αποτέλεσμα:



Διαπιστώνουμε, λοιπόν, ότι το πρόγραμμα τρέχει όπως και πριν, για όλες τις δυνατές περιπτώσεις ορισμάτων που μπορούμε να του δώσουμε.

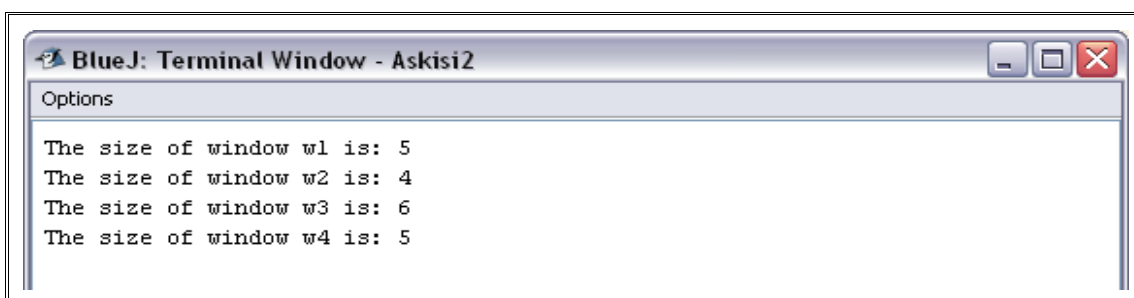
2^η Εργαστηριακή Άσκηση

3) Τρέχοντας το πρόγραμμα σύμφωνα με τις οδηγίες στο φυλλάδιο των εργαστηριακών ασκήσεων, δηλαδή μη δίνοντας κάποιο όρισμα, πήραμε τα παρακάτω αποτελέσματα:



```
BlueJ: Terminal Window - Askisi2
Options
The size of window w1 is: 5
The size of window w2 is: 4
The size of window w3 is: 6
The size of window w4 is: 5
```

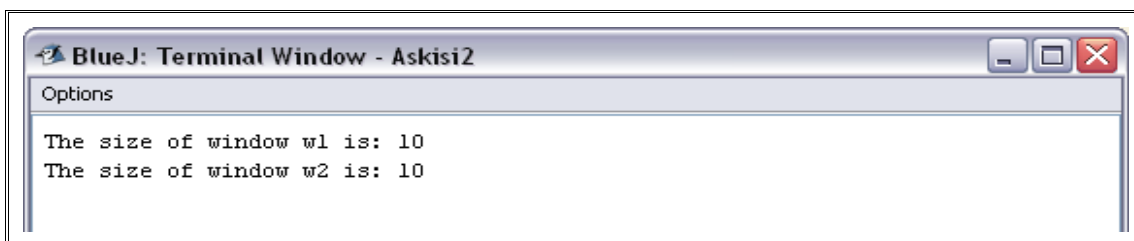
4) Χρησιμοποιώντας στον δεύτερο δημιουργό την εντολή `this.size=x;` αντί της `size=x;`, ξαναμεταφράζοντας και τρέχοντας το προγράμμα μας, πήραμε το ακόλουθο αποτέλεσμα:



```
BlueJ: Terminal Window - Askisi2
Options
The size of window w1 is: 5
The size of window w2 is: 4
The size of window w3 is: 6
The size of window w4 is: 5
```

Παρατηρούμε ότι είναι ίδιο με το αποτέλεσμα στο προηγούμενο ερώτημα της άσκησης, συνεπώς δεν υπήρξε κάποια αλλαγή μετά την προσθήκη του `this` στην εντολή. Αυτό συμβαίνει διότι κάθε instance της κλάσης `Window` που δημιουργείται έχει δική του μεταβλητή `size` ορισμένη μία φορά ως `private`, συνεπώς η απλή αναφορά σε αυτή ισοδυναμεί με την πιο ειδικευμένη αναφορά με το `this`.

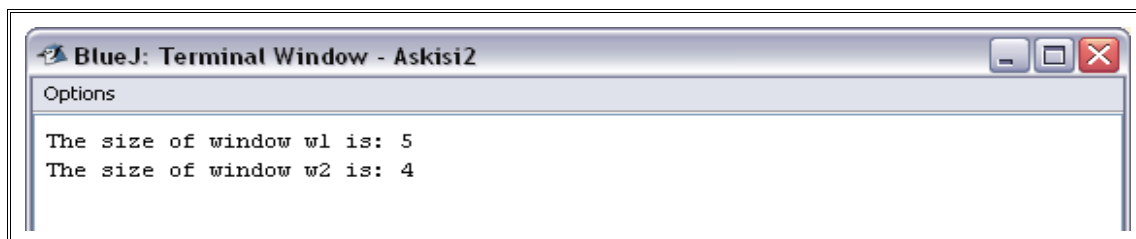
5) Έπειτα από την αλλαγή του κώδικα όπως μας υποδεικνύεται στην εκφώνηση έχουμε:



```
BlueJ: Terminal Window - Askisi2
Options
The size of window w1 is: 10
The size of window w2 is: 10
```

Παρατηρούμε ότι και για τα δύο instances εκτυπώνεται η ίδια τιμή για τη μεταβλητή `size`, παρόλο που οι constructors τους κλήθηκαν με διαφορετικά ορίσματα και διαφορετικές τιμές για τη μεταβλητή `size` κι έτσι θα περιμέναμε διαφορετικό αποτέλεσμα. Αυτό οφείλεται στο ότι κατά την κλήση της μεθόδου `getSize()` για την εκτύπωση της τιμής της μεταβλητής `size`, ουσιαστικά αναζητείται πρώτα η μεταβλητή στο εύρος της μεθόδου και ύστερα στο υπόλοιπο τμήμα του προγράμματος. Με αποτέλεσμα να εκτυπώνεται η τιμή «10», όπως αρχικοποιείται με κάθε κλήση της μεθόδου.

6) Αντικαθιστώντας την εντολή `return size` με την εντολή `return this.size` στη μέθοδο `getSize()`, έχουμε το εξής αποτέλεσμα:



Παρατηρούμε ότι σε σχέση με τα προηγούμενα αποτελέσματα είναι διαφορετικά, και μάλλον τα επιθυμητά, καθώς πλέον εκτυπώνεται όντως η αναμενόμενη τιμή της μεταβλητής `size`. Αυτό συμβαίνει επειδή με την εισαγωγή του `this` η μεταβλητή αναζητείται στον κώδικα του «αμέσως επόμενου επιπέδου» του προγράμματος, ας πούμε. Με την έννοια ότι μόλις γίνει αναφορά στη μεταβλητή `size` με το προσδιοριστικό `this`, η τιμή της αναζητείται στο κομμάτι του προγράμματος εκτός της μεθόδου, στην οποία ορίζεται και αρχικοποιείται μια δεύτερη μεταβλητή με το ίδιο όνομα.

3^η Εργαστηριακή Άσκηση

2) α. Τα αποτελέσματα μετά την εκτέλεση του προγράμματος είναι τα εξής:



```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
```

Ουσιαστικά, κατά την εκτέλεση δημιουργούνται δύο instances της κλάσης Window με διαφορετικά ορίσματα και εκτυπώνεται το μήνυμα που βρίσκεται στον κάθε constructor και η τιμή της μεταβλητής size αντίστοιχα.

β. Μετά την προσθήκη του κώδικα και την εκτέλεση του προγράμματος έχουμε τα ακόλουθα:

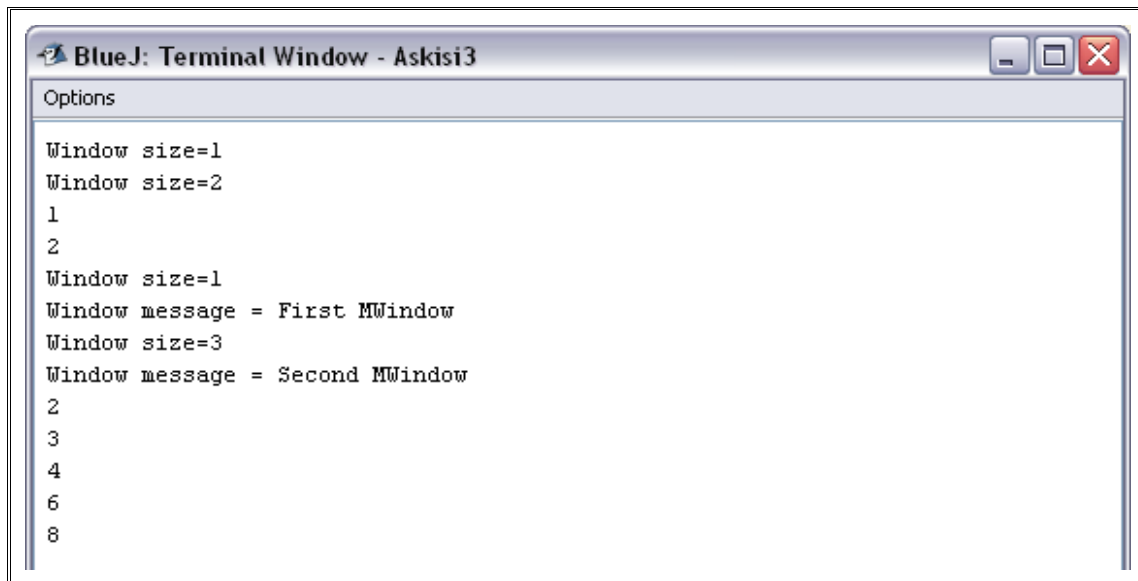


```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
Window size=1
Window message = First MWindow
Window size=3
Window message = Second MWindow
2
3
```

Παρατηρούμε ότι αφού ένα κομμάτι του κώδικα παρέμεινε ίδιο, έχουμε τα ίδια αποτελέσματα. Όσον αφορά το κομμάτι που προστέθηκε, ουσιαστικά δημιουργούνται δύο instances της κλάσης MWindow με κλήση των δύο constructors της. Αυτό που έχει ενδιαφέρον είναι η εκτύπωση των μηνυμάτων “Window size=1” και “Window size=3” πριν την εκτύπωση του “Window message = ...”. Προκαλείται επειδή σε κάθε constructor μια υποκλάσης υπάρχει μια κλήση `super()` για τον αντίστοιχο constructor της υπερκλάσης της. Έτσι, στον πρώτο constructor της MWindow υπάρχει και καλείται αλλά δε φαίνεται, ενώ στον δεύτερο φαίνεται και μάλιστα προσδιορίζεται ο δεύτερος constructor της Window, καθώς δίνεται ως όρισμα η τιμή της μεταβλητής size. Τέλος, εκτυπώνονται οι τιμές της μεταβλητής size για κάθε instance από αυτά που δημιουργήθηκαν.

γ. Προσθέτοντας πάλι ένα κομμάτι κώδικα σύμφωνα με την υπόδειξη της άσκησης έχουμε:

(εικόνα στην επόμενη σελίδα)



```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
Window size=1
Window message = First MWindow
Window size=3
Window message = Second MWindow
2
3
4
6
8
```

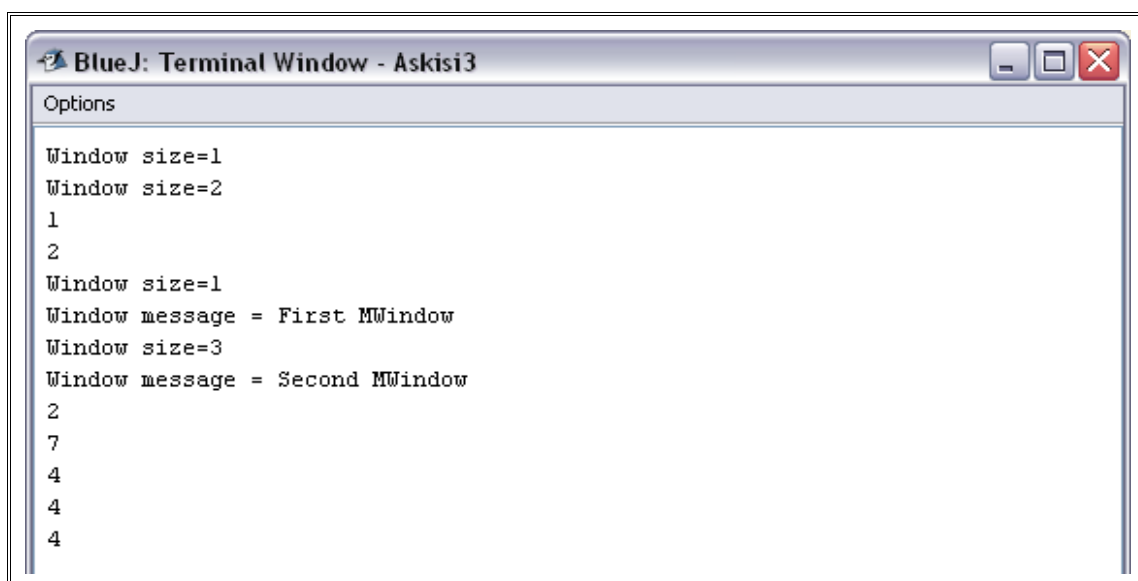
Πάλι έχουμε τα ίδια αποτελέσματα για το κομμάτι κώδικα που προϋπήρχε, οπότε ας εστιάσουμε στα τρία τελευταία νούμερα που εκτυπώθηκαν, μιας και σε αυτά αναφέρεται το κομμάτι κώδικα που προσθέσαμε.

Αρχικά, παρατηρούμε ότι γίνεται κλήση της μεθόδου `setSize()` του instance `mw1` με όρισμα «4». Αυτή η μέθοδος θέτει στην μεταβλητή `size` την τιμή «4», οπότε σωστά εκτυπώνεται στη συνέχεια η ίδια τιμή με την εντολή `System.out.println(mw1.size);`.

Στη συνέχεια, καλείται η μέθοδος `setSize()` με όρισμα «2» του ίδιου instance `mw1`. Βλέπουμε ότι αυτή με τη σειρά της καλεί τη μέθοδο `setSize()` της υπερκλάσης μέσω της εντολής `super.setSize(z);`, η οποία προσθέτει το όρισμα της αρχικής κλήσης στην προηγούμενη τιμή της μεταβλητής `size`. Άρα, αφού $4+2=6$, εκτυπώνεται ορθά «6».

Τέλος, καλείται κατευθείαν η μέθοδος `setSize()` με όρισμα «2», πάλι για το instance `mw1`, οπότε επαναλαμβάνεται η παραπάνω διαδικασία προσθήκης του ορίσματος στην τιμή της μεταβλητής. Έτσι, έχουμε $6+2=8$, άρα πάλι εκτυπώνεται ορθά το «8».

3) Κάνοντας τις αλλαγές που υποδεικνύονται στην εκφώνηση, έχουμε το παρακάτω αποτέλεσμα:



```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
Window size=1
Window message = First MWindow
Window size=3
Window message = Second MWindow
2
7
4
4
4
```

Παρατηρούμε ότι τα τέσσερα τελευταία νούμερα που εκτυπώνονται είναι διαφορετικά από την

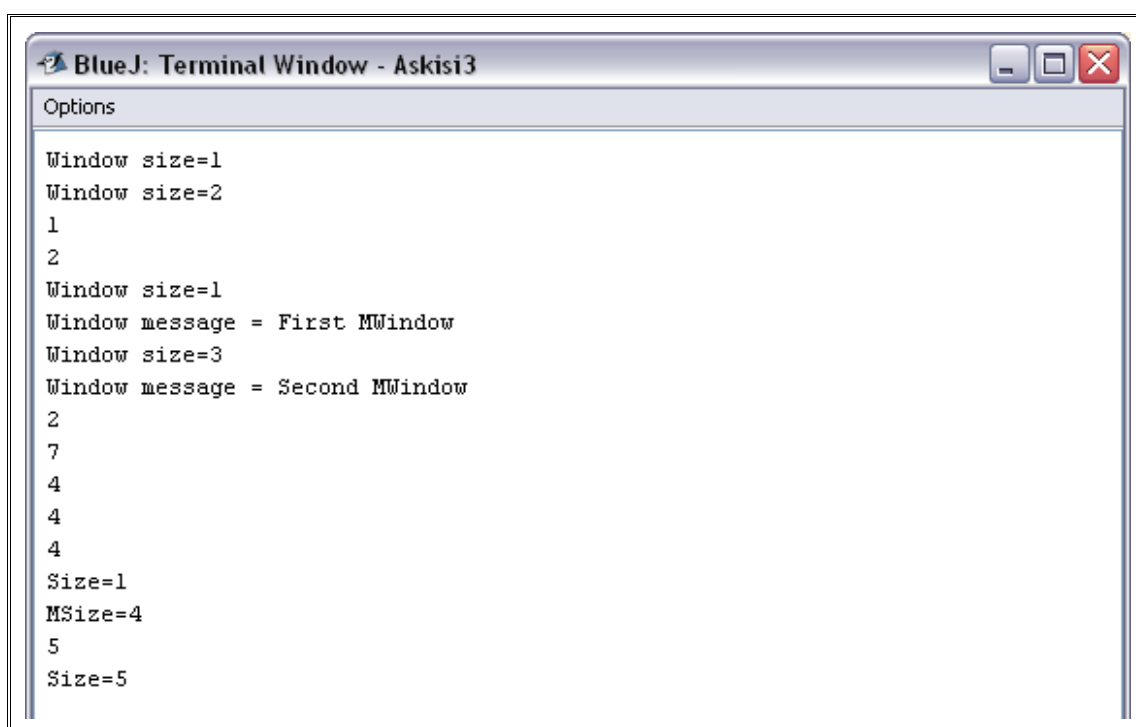
προηγούμενη εκτέλεση, κι αυτό οφείλεται σίγουρα στην προσθήκη της δήλωσης της `protected` μεταβλητής `size` μέσα στην κλάση της `MWindow`.

Αρχικά, παρατηρούμε ότι η εντολή `System.out.println(mw2.size);` εκτυπώνει την τιμή «7». Αυτό οφείλεται στο ότι αναζητείται η μεταβλητή `size` πρώτα στο χώρο της κλάσης `MWindow` και μετά στο χώρο της υπερκλάσης. Οπότε, παρόλο που φαινομενικά θέτουμε κατά την δημιουργία του instance `mw2` την τιμή της `size` «3», ουσιαστικά θέτουμε τη μεταβλητή η οποία βρίσκεται στο χώρο της `Window`, ενώ η `size` στην `MWindow` παραμένει «7», γι' αυτό και τυπώνεται αυτή στο τέλος.

Έπειτα, έχουμε την εντολή `System.out.println(mw1.size);` η οποία εκτυπώνει την τιμή «4». Καθώς προηγείται η κλήση της μεθόδου `setSize1()`, όπου η μεταβλητή `size` της `MWindow` αλλάζει από «7» σε «4», το αποτέλεσμα είναι το αναμενόμενο. Στη συνέχεια όμως, παρόλο που έχουμε κλήσεις των μεθόδων `setSize2()` και `setSize()` οι οποίες φαίνεται ότι αλλάζουν την τιμή της `size`, ουσιαστικά έχουμε το ίδιο αποτέλεσμα στις εκτυπώσεις, κι αυτό γιατί οι κλήσεις των μεθόδων αυτών αλλάζουν την τιμή στη μεταβλητή `size` που βρίσκεται στον χώρο της `Window`, ενώ οι εντολές εκτύπωσης αναφέρονται σε αυτή που βρίσκεται στον χώρο της `MWindow`. Άρα, αφού δεν αλλάζει η τιμή της τελευταίας, το αποτέλεσμα της εκτύπωσης παραμένει «4».

Σε σχέση με το προηγούμενο ερώτημα δεν χρειάζεται κάποια αναθεώρηση στην εξήγηση των αποτελεσμάτων, αλλά στην ουσία με το παρόν ερώτημα ξεκαθαρίζει η λειτουργία και η εμβέλεια των μεταβλητών μεταξύ των κλάσεων, υποκλάσεων και υπερκλάσεων.

4) Προσθέτοντας το κομμάτι κώδικα που μας δίνεται έχουμε:



```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
Window size=1
Window message = First MWindow
Window size=3
Window message = Second MWindow
2
7
4
4
4
Size=1
MSize=4
5
Size=5
```

Προσθέσαμε τέσσερις κλήσεις μεθόδων οι οποίες εκτυπώνουν αντίστοιχα αποτελέσματα. Ας εξετάσουμε αναλυτικά κάθε μία για να δούμε τη λειτουργία τους.

Αρχικά, έχουμε την εντολή `w1.printSize();`, η οποία εκτυπώνει την τιμή της μεταβλητής `size` του instance `w1`, που είναι όντως ίση με «1», μιας και κλήθηκε ο πρώτος constructor της `Window` κατά τη δημιουργία του `w1`, όπου τίθεται η `size` ίση με «1».

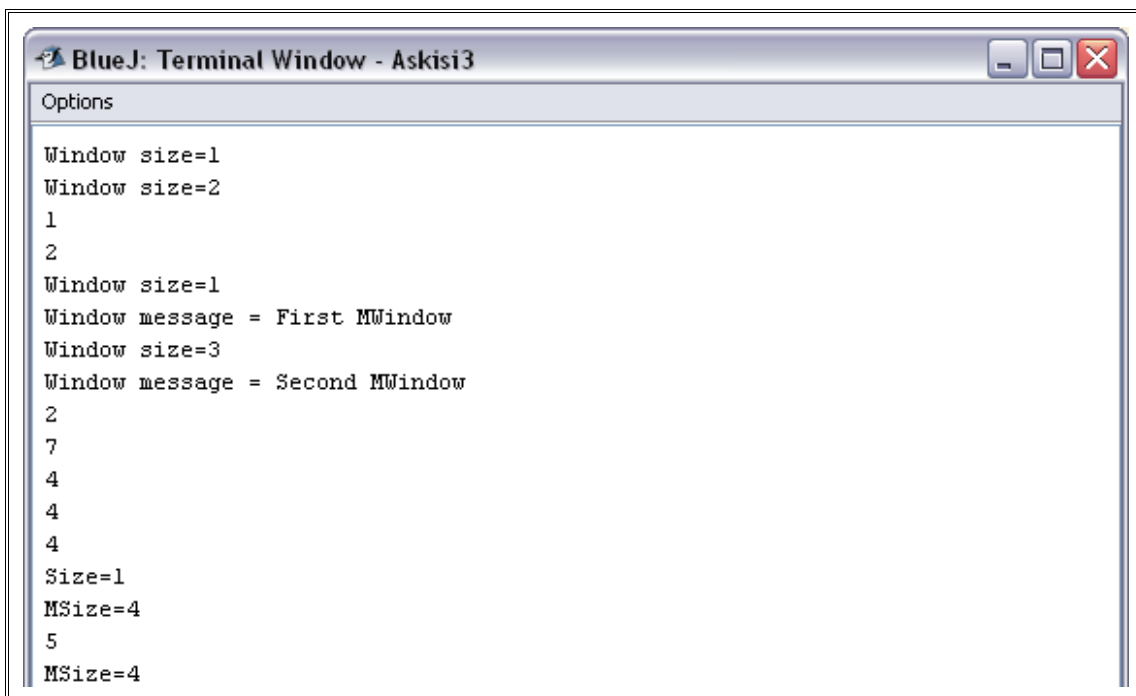
Έπειτα, έχουμε την εντολή `mw1.printSize();`, η οποία εκτυπώνει την τιμή της μεταβλητής `size` (του instance `mw1`) που βρίσκεται στο χώρο της κλάσης `MWindow` και ισούται με «4» για τους λόγους που εξηγήσαμε στο προηγούμενο ερώτημα.

Μετέπειτα, έχουμε την εντολή `mw1.printSize1();` που αναφέρεται πάλι στο instance `mw1`.

Εδώ οφείλουμε να αναλύσουμε λίγο καλύτερα το γιατί εκτυπώνεται η τιμή «5». Αν δούμε τα πράγματα από την αρχή της δημιουργίας της `mw1`, θα παρατηρήσουμε πως με την κλήση του constructor η μεταβλητή `size` στην `MWindow` παίρνει την τιμή «2» ενώ η μεταβλητή `size` στην `Window` παίρνει την τιμή «1». Στη συνέχεια, με την κλήση της μεθόδου `setSize1()` η μεταβλητή `size` της `MWindow` παίρνει την τιμή «4». Με την κλήση της μεθόδου `setSize2()`, καλείται ουσιαστικά η μέθοδος `setSize()` της `Window`, η οποία και προσθέτει στην τιμή της μεταβλητής `size` (που ήταν ίση με «1») το όρισμα της, δηλαδή $1+2=3$, οπότε η μεταβλητή `size` της `Window` έχει πλέον την τιμή «3». Αντίστοιχα, με την άμεση κλήση της `setSize()` αυξάνεται πάλι η τιμή της, άρα $3+2=5$, και η τιμή της πλέον είναι «5», όπως και εκτυπώνεται στο τέλος από την μέθοδο `printSize1()`. Κι αυτό γιατί η μέθοδος αυτή αναφέρεται στην μεταβλητή `size` της υπερκλάσης μέσω της `"System.out.println (super.size);"`.

Τέλος, έχουμε την εντολή `"mw1.printSize2();"`, η οποία καλεί τη μέθοδο `printSize2()`, που κι αυτή με τη σειρά της καλεί την μέθοδο της υπερκλάσης `printSize()`. Γι' αυτό τυπώνεται η τιμή της μεταβλητής `size` που βρίσκεται στην υπερκλάση `Window` (και ισούται με «5», όπως εξηγήσαμε πριν).

5) Αλλάζοντας τις μεθόδους σύμφωνα με τον κώδικα που μας δίνεται, έχουμε τα εξής:



```
BlueJ: Terminal Window - Askisi3
Options
Window size=1
Window size=2
1
2
Window size=1
Window message = First MWindow
Window size=3
Window message = Second MWindow
2
7
4
4
4
Size=1
MSize=4
5
MSize=4
```

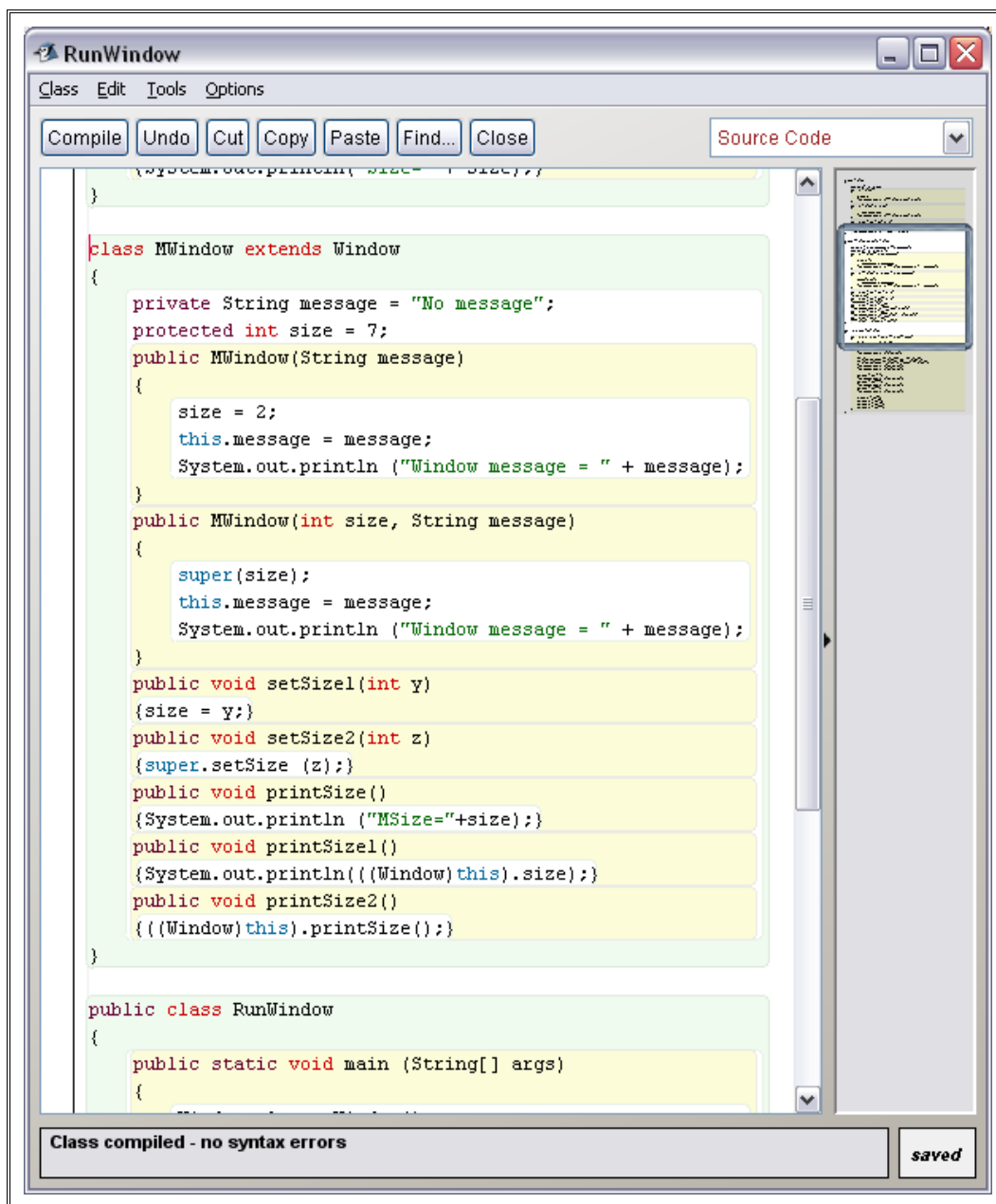
Σύμφωνα με τη θεωρία (που παρατίθεται και στις αναφερόμενες στην εκφώνηση σελίδες του βιβλίου του Κ. Θραμπουλίδη, «Αντικειμενοστραφής Προγραμματισμός JAVA»), οι εκφράσεις `"((Window)this).size;"` και `"((Window)this).printSize();"` αποτελούν ουσιαστικά άμεση μετατροπή (ή αλλιώς casting) της αναφοράς `this` στην αντίστοιχη υπερκλάση ή πρόγονο κλάση (στην περίπτωσή μας, τη `Window`), και στη συνέχεια αναφορά στη μεταβλητή `size` και στην μέθοδο `printSize()`.

Στην αναφορά στη μεταβλητή `size` δεν αλλάζει κάτι, μιας και έχουμε μόνο μία πρόγονο κλάση. Επομένως, η έκφραση `"super.size;"` είναι ισοδύναμη με την `"((Window)this).size;"`, άρα και το αποτέλεσμα σωστά παραμένει το ίδιο.

Στην αναφορά στη μέθοδο `printSize()` με αυτόν τον τρόπο όμως, παρόλο που λειτουργεί αντίστοιχα με την προηγούμενη αναφορά, διαφέρει στο ότι οι μέθοδοι υπερκαλύπτονται και δεν επικαλύπτονται, κάτι που σημαίνει πως η αναφορά σε υπερκαλυπτόμενη μέθοδο δε μπορεί να γίνει με τον ίδιο τρόπο όπως σε μια επικαλυπτόμενη μεταβλητή. Έτσι, στην αναφορά για τη μέθοδο `printSize()` στη συγκεκριμένη περίπτωση, οφείλαμε να προσδιορίσουμε το στιγμιότυπο (π.χ. το `w1`) της κλάσης `Window` στο οποίο αναφερόμαστε, ώστε να είναι επιτυχής η αναφορά στη μέθοδο, περίπου ως

“(Window)w1).printSize();”. Εδώ όμως βλέπουμε πως αποτυγχάνει η αναφορά, οπότε και καλείται ουσιαστικά η μέθοδος της κλάσης MWindow, γι' αυτό και εκτυπώνεται το αποτέλεσμα αυτό.

6) Αντιγράφοντας όλους τους επιμέρους κώδικες σε μία κλάση, όπως μας υπαγορεύεται και στην εκφώνηση, έχουμε την ακόλουθη εικόνα για την κλάση RunWindow:



Έπειτα από εκτέλεση του παραπάνω προγράμματος έχουμε οντως τα ίδια αποτελέσματα με την εκτέλεση του προγράμματος όπως ήταν πριν τη συνένωση των κομματιών κώδικα.

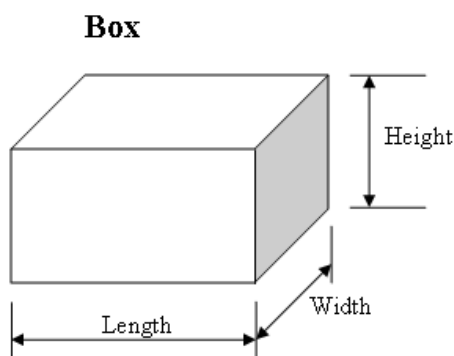
Μέρος Α' 1^ο Σετ Ασκήσεων

Οι απαντήσεις δίνονται χωρίς γραπτή τεκμηρίωση, σύμφωνα με την ίδια την εκφώνηση.

- 1) C, E
- 2) C
- 3) D
- 4) B
- 5) A
- 6) B

Μέρος Β' 1^ο Σετ Ασκήσεων

1) Η μέθοδος `calculate()`, δεδομένου ότι θεωρητικά έχουμε να κάνουμε με ένα ορθογώνιο παραλληλεπίπεδο (ή κουτί), υπολογίζει την συνολική επιφάνεια που καταλαμβάνουν οι πλευρές του ορθογωνίου. Αυτό γίνεται αντιληπτό αν εξετάσουμε και την τρισδιάστατη απεικόνισή του, υπολογίσουμε το εμβαδόν κάθε πλευράς του και τα αθροίσουμε όλα μαζί:



(όπου *length*=ύψος, *width*=πλάτος, *height*=ύψος)

$$\begin{aligned}\text{Συνολική Επιφάνεια} &= 2 * \text{μήκος} * \text{πλάτος} + 2 * \text{μήκος} * \text{ύψος} + 2 * \text{ύψος} * \text{πλάτος} \\ &= 2 * (\text{μήκος} * \text{πλάτος} + \text{μήκος} * \text{ύψος} + \text{ύψος} * \text{πλάτος})\end{aligned}$$

2) Ένας constructor σύμφωνα με τα πρότυπα της εκφώνησης είναι ο ακόλουθος:

```
Box (double h, double w, double l){
    height=h;
    width=w;
    length=l;
}
```

3) Μια μέθοδος `volume()` που θα υπολογίζει τον όγκο του ορθογωνίου θα μπορούσε να είναι η:

```
double volume(){
    return height*width*length;
}
```

4) & 5) Μια μέθοδος `toString()` η οποία θα εκτυπώνει το αποτέλεσμα των παραπάνω μεθόδων θα μπορούσε να ήταν η ακόλουθη:

```
public String toString(){
    double calc=calculate();
    double vol=volume();
    String s="Calculation " + calc + ", Volume " + vol;
    System.out.println(s);
    return s;
}
```

Το πρόβλημα που δημιουργήθηκε ήταν στη δήλωση της μεθόδου. Αρχικά, αν τη δηλώσουμε ως `void` προκύπτει σφάλμα, καθώς υπάρχει ήδη ορισμένη `public void toString()` στην κλάση

Object, υποκλάση της οποίας είναι αυτομάτως όποια κλάση δημιουργούμε. Έτσι, πρέπει να ορίσουμε μια μέθοδο `toString()` διαφορετικού τύπου, όπως π.χ. `String`, για να μην υπάρχει επικάλυψη των μεθόδων, σύμφωνα και με την παραίτηση του 5^{ου} ερωτήματος.

6) Σύμφωνα με τα ζητούμενα του ερωτήματος, ο κώδικας για την `MyBox` θα είναι ο ακόλουθος:

```
public class MyBox extends Box{
    double side;

    public MyBox(double h, double w, double l){
        super(h, w, l);

        if(h!=w || w!=l || h!=l){
            System.out.println("ERROR: this is not a cube.");
        }else if(h<0 || w<0 || h<0){
            System.out.println("ERROR: sides must be greater than 0.");
        }else if((w==h)&&(h==l)){
            side=h;
        }
    }
}
```

7) Οι μέθοδοι `calculate()` και `volume()` θα μπορούσαν να τροποποιηθούν ως κάτωθι και να συμπεριληφθούν στην κλάση της `MyBox`:

```
double calculate(){
    return 6*(side*side);
}

double volume(){
    return side*side*side;
}
```

8) Η ζητούμενη μέθοδος `toString()` για την εκτύπωση των αποτελεσμάτων της `calculate()` και της `volume()` θα μπορούσε να είναι η εξής:

```
public String toString(){
    double calc=calculate();
    double vol=volume();
    String s="Cube Calculation " + calc + ", Volume " + vol;
    System.out.println(s);
    return s;
}
```

9) & 10) Δεδομένης της συσχέτισης μεταξύ των δύο ερωτημάτων, ο παρακάτω κώδικας απαντάει και στα δύο ταυτόχρονα:

```
public class Spirtokouto extends Box{
    double weight;

    public Spirtokouto(double h, double w, double we, double l){
        super(h, w, l);
        weight=we;
    }
}
```

```
}
```

11) Η ζητούμενη μέθοδος θα μπορούσε να είναι η ακόλουθη:

```
public String toString(){
    double calc=super.calculate();
    double vol=super.volume();
    String s="Matchbox Calculation " + calc + ", Volume " + vol
    + ", Weight " + weight;
    System.out.println(s);
    return s;
}
```

12) Η κλάση RunMe που θα περιλαμβάνει τη main θα ήταν η εξής:

```
public class RunMe{
    public static void main(String args[]){
        Box box_1=new Box(2.5, 3.0, 4.0);
        MyBox box_2=new MyBox(3.0, 3.0, 3.0);
        Spirtokouto box_3=new Spirtokouto(2.0, 4.0, 10.0, 3.0);
        box_1.toString();
        box_2.toString();
        box_3.toString();
    }
}
```

Τα αποτελέσματα έπειτα από την εκτέλεσή της είναι τα ακόλουθα:

