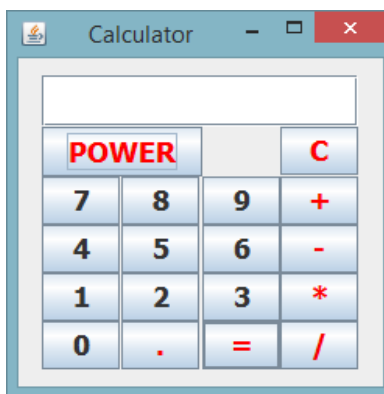


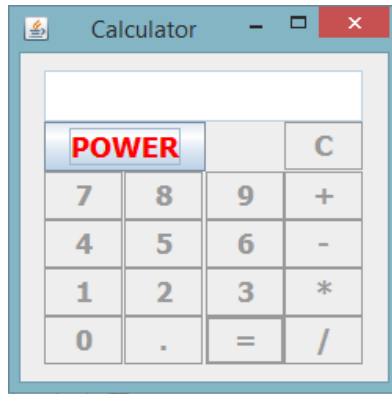
Εργαστήριο 2 - Άσκηση - Ανάλυση

Εκφώνηση: Δημιουργήστε την εφαρμογή Αριθμήτηριο σε Java Swing με χρήση NetBeans ακολουθώντας τις παρακάτω οδηγίες.

1. Η εφαρμογή θα σχεδιασθεί σε ένα εξωτερικό υποδοχέα JFrame, ο οποίος θα έχει τίτλο «Calculator», διάταξη **GridBagLayout** και θα οριστεί με την έξοδο του να κλείνει και η εφαρμογή.
2. Τοποθετήστε στο JFrame ένα TextField και 18 Buttons.
3. Τοποθετήστε τα συστατικά όπως φαίνονται στην παρακάτω εικόνα με χρήση του εργαλείου 'Customize Layout' (δεξί click πάνω στο φόντο).
4. Αλλάξτε τις ιδιότητες του TextField ως εξής:
 - Variable name: calcbox
 - Font: bold 24 μέγεθος.
 - Horizontal Alignment: Right
5. Αλλάξτε τις ιδιότητες των 18 Buttons ως εξής:
 - Font: bold 18 μέγεθος
 - το χρώμα το κουμπιών να ταιριάζει με την παρακάτω εικόνα:



6. Να προγραμματιστεί το κουμπί POWER ώστε να ενεργοποιεί ή απενεργοποιεί όλα τα άλλα συστατικά με χρήση της μεθόδου `setEnabled(Boolean)`. Αρχικά όλα τα συστατικά να είναι απενεργοποιημένα όπως φαίνονται στην παρακάτω εικόνα:



7. Να προγραμματιστούν τα δέκα κουμπιά αριθμών ώστε να προσθέτουν τον αντίστοιχο αριθμό κάθε κουμπιού δεξιά του περιεχομένου του **calcbox**.
8. Να προγραμματιστεί το κουμπί «.» ώστε να μπαίνει η υποδιαστολή στο **calcbox**. Προσοχή να μην μπαίνει δεύτερη υποδιαστολή στον ίδιο αριθμό.
9. Να προγραμματιστούν τα τέσσερα κουμπιά των πράξεων για να υλοποιούν τις αντίστοιχες πράξεις.
10. Να προγραμματιστεί το κουμπί ίσον ώστε να δίνει αποτέλεσμα:
 - 0, αν πατηθεί το ίσον πριν την εισαγωγή κάποιου αριθμού,
 - ο πρώτος αριθμός που εισήχθη, εάν μετά το πάτημα ενός συμβόλου πράξης πατηθεί το ίσον χωρίς εισαγωγή άλλου αριθμού,
 - μήνυμα λάθους, αν σε διαίρεση ο δεύτερος αριθμός είναι το 0,
 - το αποτέλεσμα της πράξης σε κάθε άλλη περίπτωση.
11. Να προγραμματιστεί το κουμπί C ώστε να αδειάζει το περιεχόμενο του **calcbox** και να μηδενίζει τις μεταβλητές που θα χρησιμοποιήσετε για τις πράξεις.
12. Να προγραμματιστεί το πλήκτρο <ENTER> στο συστατικό κειμένου ώστε να έχει την ίδια λειτουργία με το ίσον.

Σημείωση: Μέθοδοι που πιθανώς θα σας φανούν χρήσιμες:

- `TextField.setText(string);`
- `TextField.getText();`
- `JComponent.setEnabled(Boolean);`
- `JComponent.isEnabled();`
- `Πεδίο_String.isEmpty();`
- `String.valueOf(double);`
- `Double.valueOf(string);`
- `Πεδίο_String.contentEquals(string)`
- `getRootPane().setDefaultButton(action);`

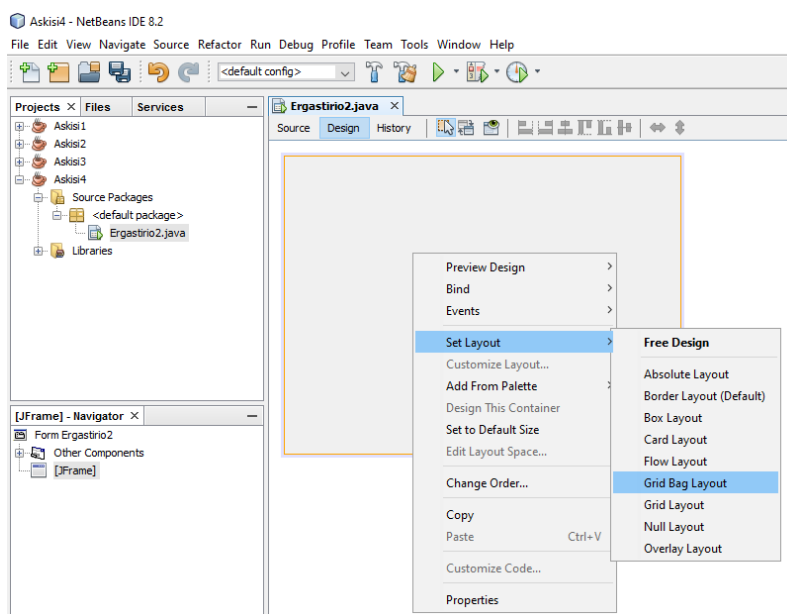
Ανάλυση και εξήγηση λύσης

Βήμα 1: Δημιουργούμε ένα νέο project στο NetBeans με όνομα Askisi4.

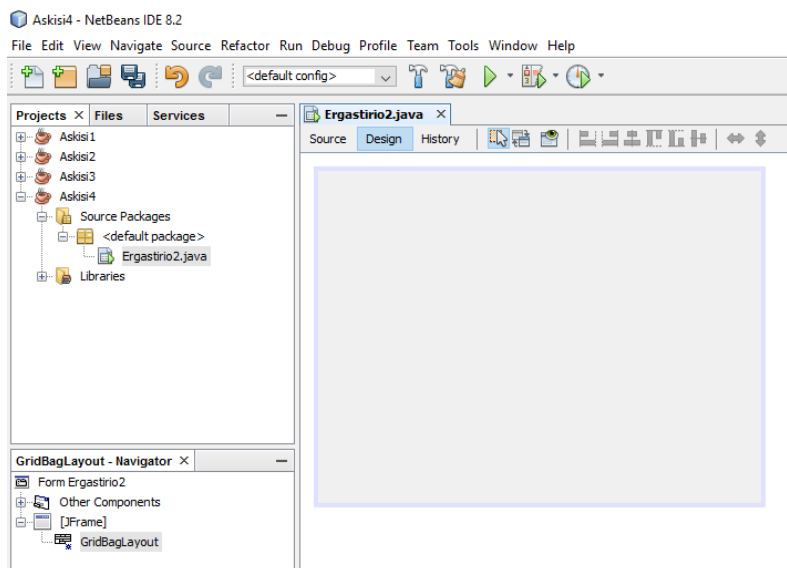
Βήμα 2: Δημιουργούμε ένα νέο Java αρχείο τύπου JFrame Form με όνομα Ergastirio2.

Βήμα 3: Ορισμός Ιδιοτήτων στο JFrame. Ορίζουμε ως τίτλο του JFrame το «Calculator».

Βήμα 4: Ορίζουμε η διάταξη του JFrame να είναι GridBagLayout. Για να ορίσουμε τη διάταξη ανοίγουμε το μενού του JFrame επιλέγοντας το δεξί πλήκτρο του ποντικιού πάνω στο JFrame. Στη συνέχεια επιλέγουμε το υπομενού «Set Layout» και τη διάταξη «GridBagLayout» - Εικόνα 1. Όταν ορίσουμε τη διάταξη, αυτή θα εμφανιστεί στο *Navigator* κάτω ακριβώς από το JFrame – Εικόνα 2.

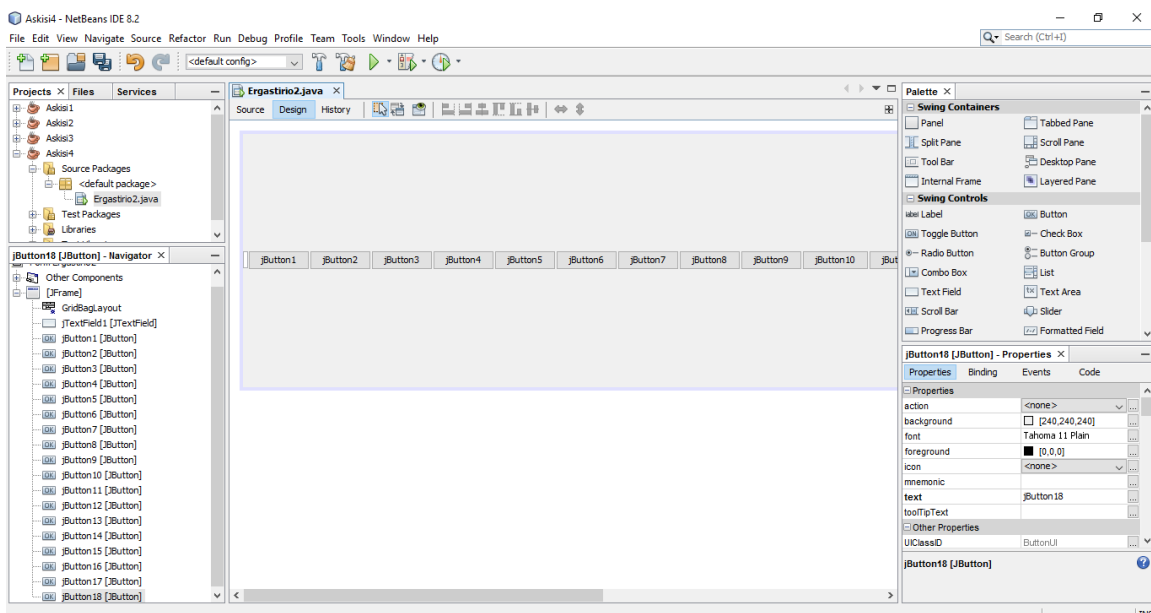


Εικόνα 1: Ορισμός διάταξης GridBagLayout



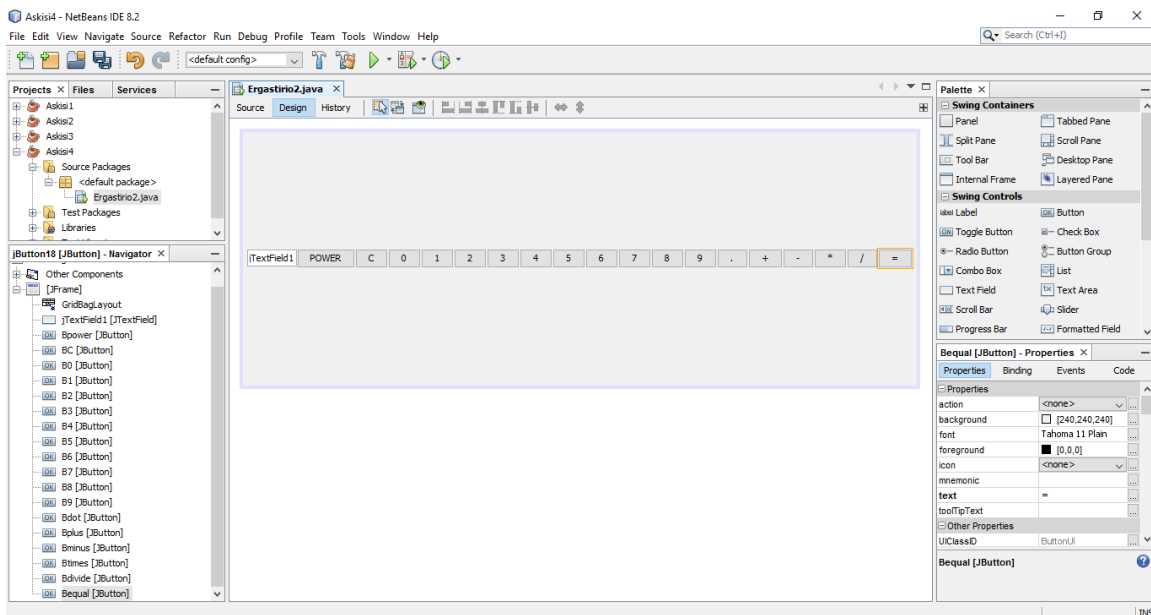
Εικόνα 2: Στο Navigator παρουσιάζεται η διάταξη που ορίσαμε

Βήμα 5: Τοποθετούμε ένα TextField και 18 Buttons με drag & drop - Εικόνα 3.



Εικόνα 3: Τοποθετήθηκε ένα Textfield και δεκαοχτώ Button

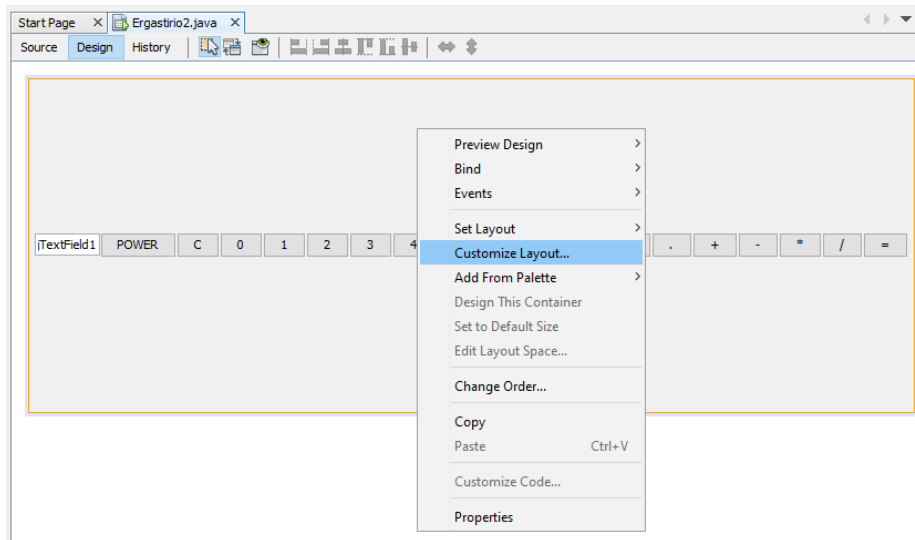
Βήμα 6: Αλλάζουμε το κείμενο που εμφανίζουν τα κουμπιά σύμφωνα με την εκφώνηση, δηλαδή ‘POWER’, ‘C’, ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘.’, ‘+’, ‘-’, ‘*’, ‘/’ και ‘=’ και τα ονόματα των μεταβλητών τους αντίστοιχα σε ‘Bpower’, ‘BC’, ‘B0’, ‘B1’, ‘B2’, ‘B3’, ‘B4’, ‘B5’, ‘B6’, ‘B7’, ‘B8’, ‘B9’, ‘Bdot’, ‘Bplus’, ‘Bminus’, ‘Btimes’, ‘Bdivide’ και ‘Bequal’.



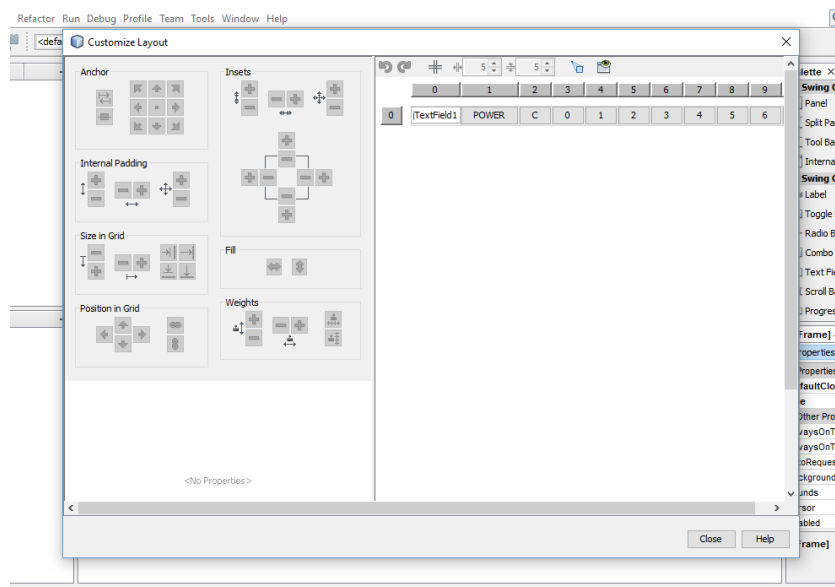
Εικόνα 4: Ορίστηκαν οι ιδιότητες text και variable name για τα δεκαοχτώ κουμπιά

Βήμα 7: Για να τοποθετήσουμε τα συστατικά στη μορφή που ζητάει η εκφώνηση θα χρησιμοποιήσουμε το εργαλείο “Customize Layout”. Για να χρησιμοποιήσουμε το εργαλείο “Customize Layout” θα πρέπει να το «ανοίξουμε» με δεξί click πάνω στο JFrame, δηλαδή στον

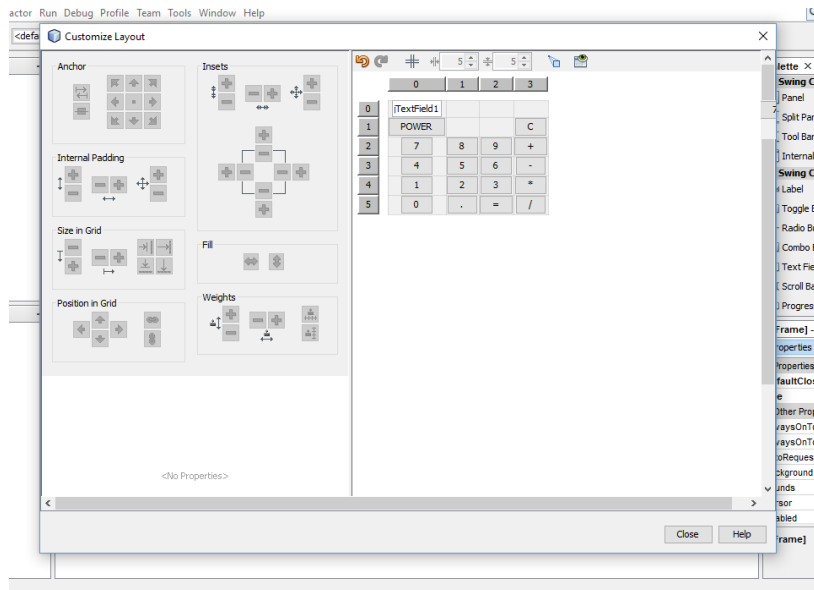
καμβά (Εικόνα 5). Στην Εικόνα 6 παρουσιάζεται το εργαλείο με την αρχική τοποθέτηση των συστατικών. Στη συνέχεια με drag & drop μετακινούμε τα συστατικά στη θέση τους, όπως φαίνεται στην Εικόνα 7. Για να καταλάβει το συστατικό POWER δύο κελιά επιλέγουμε το συστατικό και στη συνέχεια πατάμε το κουμπί *Horizontal Size in Grid* (Εικόνα 8). Για να απλωθεί το συστατικό σε όλο το πλάτος των κελιών επιλέγουμε το συστατικό και στη συνέχεια πατάμε το κουμπί *Horizontal Fill* (Εικόνα 9). Χρησιμοποιούμε το *Horizontal Size in Grid* στο συστατικό jTextField1, ώστε να καταλάβει 4 κελιά. Εφαρμόζουμε το *Horizontal Fill* σε όλα τα συστατικά. Στην Εικόνα 10, φαίνεται η τελική τοποθέτηση των συστατικών μετά την εφαρμογή όλων των παραπάνω. Κλείνουμε το εργαλείο για να προχωρήσουμε στο επόμενο βήμα.



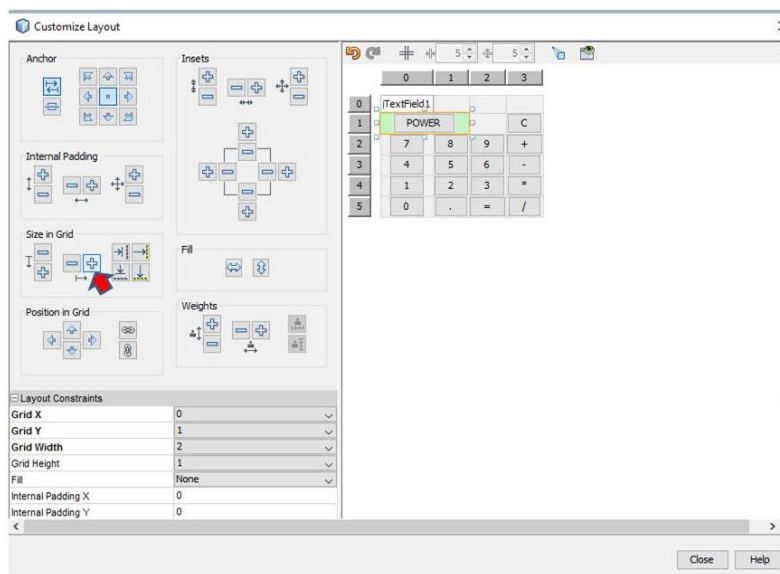
Εικόνα 5: Άνοιγμα του εργαλείου Customize Layout



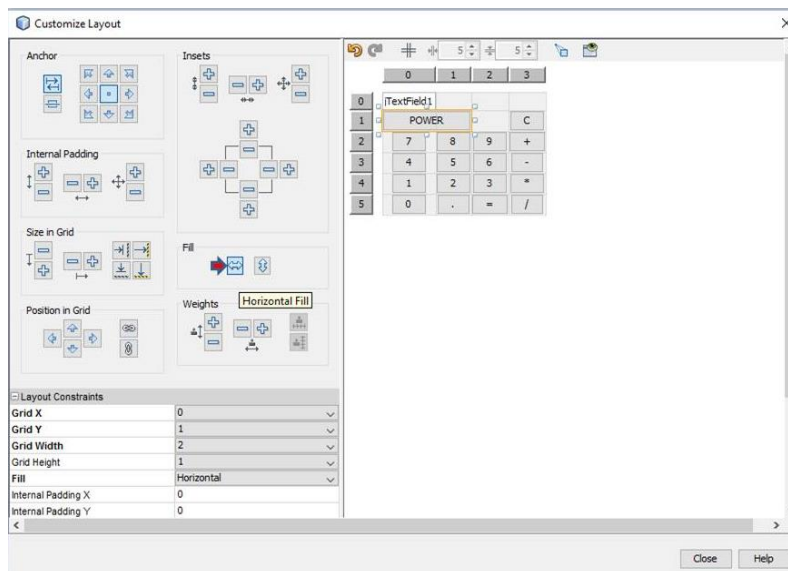
Εικόνα 6: Το εργαλείο Customize Layout – Τα συστατικά στην αρχική τους θέση



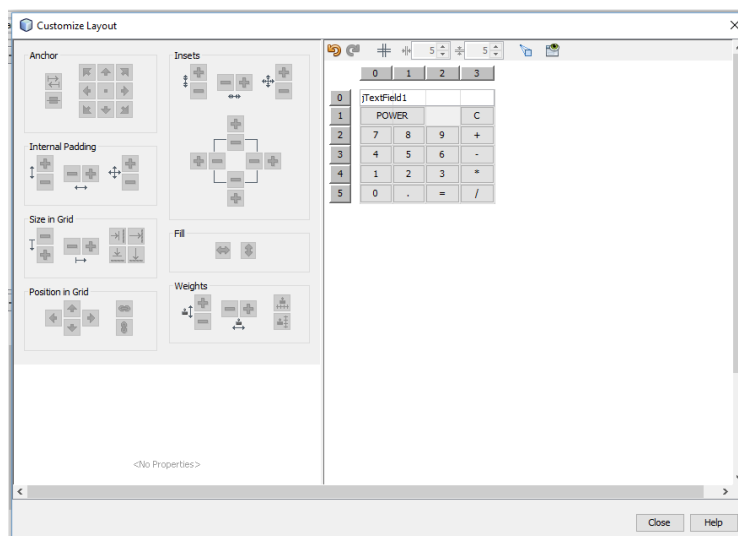
Εικόνα 7: Τα συστατικά στην τελική τους θέση



Εικόνα 8: Χρήση του Horizontal Size in Grid στο συστατικό POWER

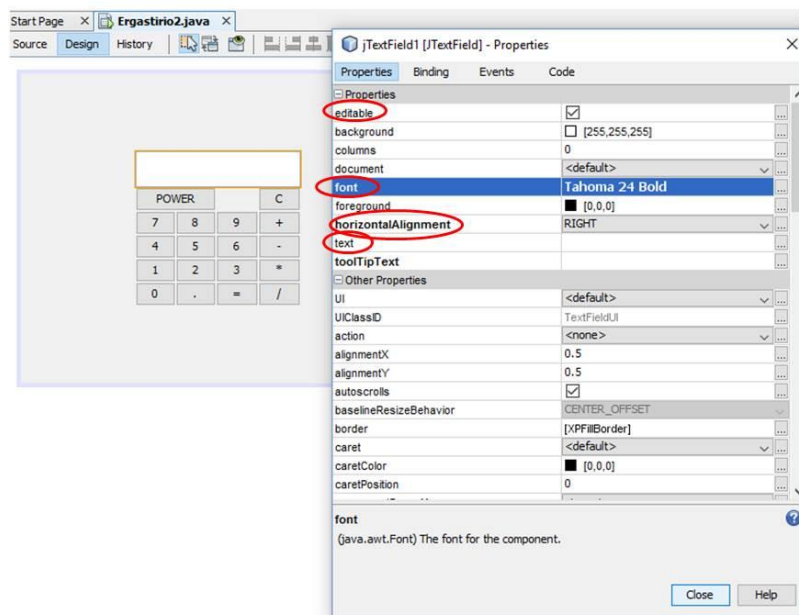


Εικόνα 9: Χρήση του Horizontal Fill στο συστατικό POWER



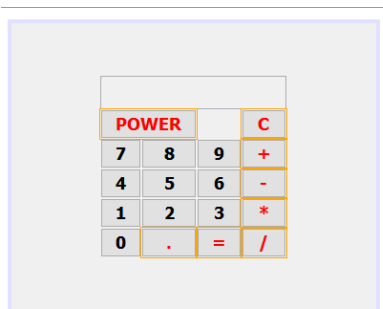
Εικόνα 10: Η τελική τοποθέτηση των συστατικών

Βήμα 8: Αλλάζουμε το όνομα μεταβλητής του `jTextField1` σε `calcbbox`. Στη συνέχεια αλλάζουμε τρεις ιδιότητες του `calcbbox` (Εικόνα 11). Την ιδιότητα `font` για να ορίσουμε μέγεθος γραμματοσειράς 24 και στυλ `bold`, την ιδιότητα `horizontalAlignment` σε `RIGHT` για δεξιά στοίχιση του περιεχομένου του `TextField` και την ιδιότητα `text` σε κενό – σβήνουμε το περιεχόμενο. Σε περίπτωση που δεν θέλουμε να μπορεί ο χρήστης να γράψει αριθμούς στο `TextField` με το πληκτρολόγιο, ορίζουμε ανενεργή την ιδιότητα `editable`.



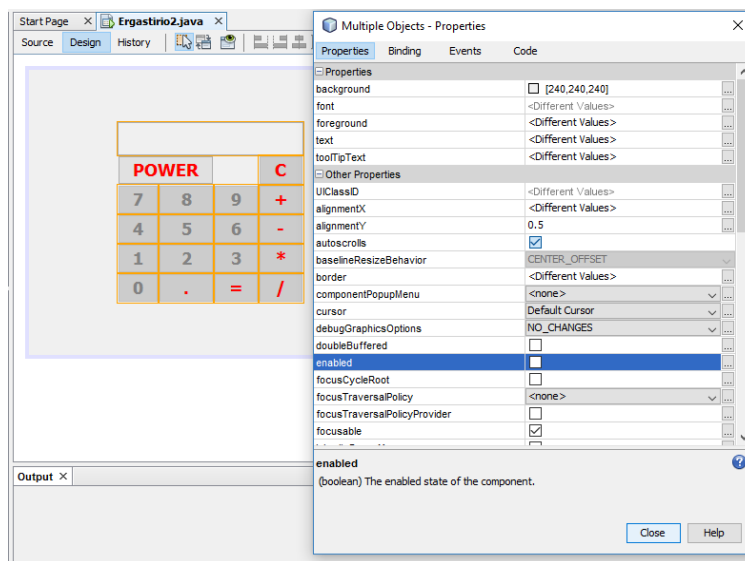
Εικόνα 11: Ορισμός ιδιοτήτων για το TextField

Βήμα 9: Επιλέγουμε τα 18 JButton και αλλάζουμε την ιδιότητα *font* όπως αναφέρεται στην εκφώνηση – μέγεθος 18, στυλ bold. Επιλέγουμε τα JButton που σύμφωνα με την εκφώνηση πρέπει να γίνουν κόκκινα, και αλλάζουμε την ιδιότητα *foreground* σε κόκκινο. Στην Εικόνα 12, παρουσιάζεται πως διαμορφώνεται η εικόνα της εφαρμογής μετά από τις παραπάνω αλλαγές.



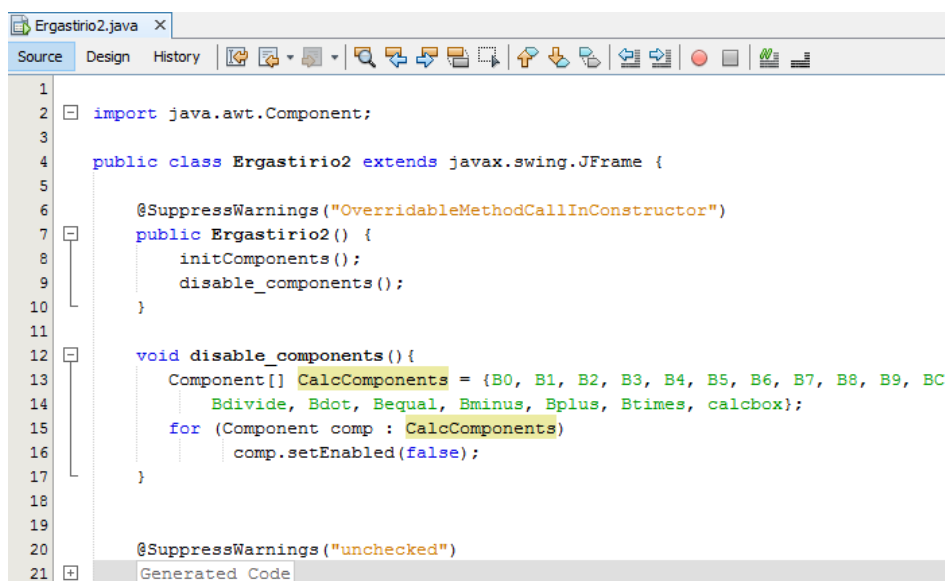
Εικόνα 12: Η εικόνα της εφαρμογής μετά την εφαρμογή του βήματος 9.

Βήμα 10: Σε αυτό το βήμα θα φροντίσουμε όλα τα συστατικά του καμβά εκτός του κουμπιού Power να «ξεκινούν» απενεργοποιημένα. Αυτό γίνεται ορίζοντας την τιμή false στην ιδιότητα *enabled* όλων των συστατικών - Εικόνα 13.



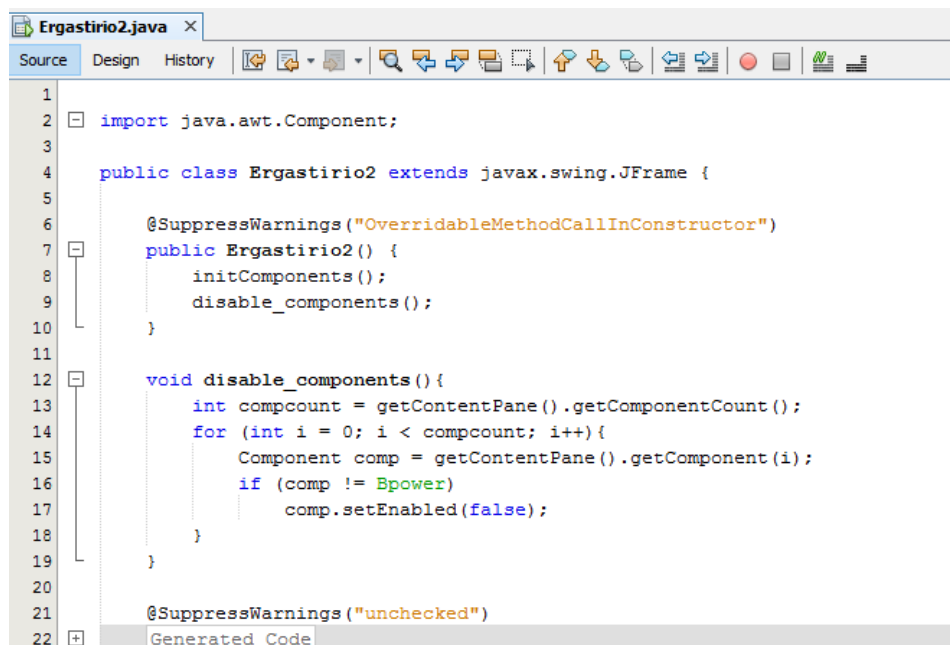
Εικόνα 13: Απενεργοποίηση των συστατικών

Εναλλακτικά, ένας δεύτερος τρόπος που μπορούμε να πετύχουμε το ίδιο αποτέλεσμα με κώδικα, φαίνεται στην Εικόνα 14. Αρχικά, καλούμε στον δομητή της τάξης, υποχρεωτικά μετά τη μέθοδο *initComponents()*, τη δική μας μέθοδο *disable_components()*. Η δήλωση της μεθόδου δίνεται στις γραμμές 12-17. Η μέθοδος δηλώνεται ως *void*. Στη γραμμή 13-14 ορίζεται ένας πίνακας από *Component* – είναι απαραίτητη η εισαγωγή της βιβλιοθήκης *java.awt.Component* (γραμμή 2). Προφανώς, στον πίνακα θα δηλώσουμε όλα τα συστατικά του καμβά εκτός του *Brower*, το οποίο είναι πάντα ενεργό. Στη συνέχεια στις γραμμές 15-16 ορίζεται η ιδιότητα *enabled* στη τιμή *false* με τη μέθοδο *setEnabled(Boolean)* για όλα τα συστατικά του πίνακα *CalcComponents*.



Εικόνα 14: Απενεργοποίηση των συστατικών με κώδικα

Ένας τρίτος τρόπος, πάλι με κώδικα φαίνεται στην Εικόνα 15. Η διαφορά από τον προηγούμενο τρόπο είναι ότι ο πίνακας των συστατικών δημιουργείται δυναμικά. Έτσι, στη γραμμή 13, αποθηκεύεται στη μεταβλητή *compcount* ο αριθμός των συστατικών που υπάρχουν στον καμβά μας με χρήση της μεθόδου *getComponentCount()*. Η *getContentPane()* μέθοδος επιστρέφει τον τρέχον υποδοχέα. Για το παράδειγμά μας είναι ο καμβάς, το *JFrame*. Στη συνέχεια, γραμμές 14-18, σε ένα κόμβο *for*, με την *getContentPane().getComponent(i)* διαβάζει τη μεταβλητή κάθε συστατικού του καμβά, την αποθηκεύει στην μεταβλητή *comp*, ελέγχει αν δεν είναι το *Brower*, και απενεργοποιεί το αντίστοιχο συστατικό.



```

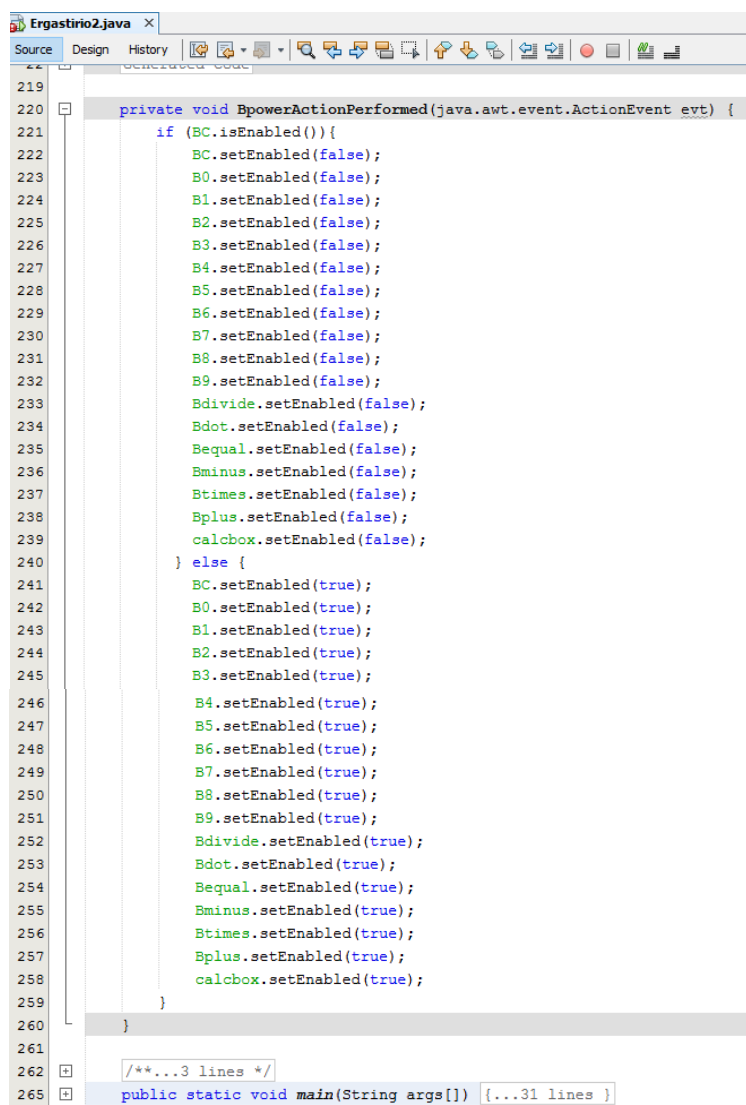
1
2 import java.awt.Component;
3
4 public class Ergastirio2 extends javax.swing.JFrame {
5
6     @SuppressWarnings("OverrideMethodCallInConstructor")
7     public Ergastirio2() {
8         initComponents();
9         disable_components();
10    }
11
12    void disable_components() {
13        int compcount = getContentPane().getComponentCount();
14        for (int i = 0; i < compcount; i++) {
15            Component comp = getContentPane().getComponent(i);
16            if (comp != Brower)
17                comp.setEnabled(false);
18        }
19    }
20
21    @SuppressWarnings("unchecked")
22    Generated Code

```

Εικόνα 15: Δημιουργία του πίνακα συστατικών δυναμικά

Βήμα 11: Το κουμπί *Brower* σύμφωνα με την εκφώνηση θα πρέπει να αλλάζει κατάσταση στα υπόλοιπα συστατικά του καμβά. Αυτό σημαίνει ότι αν τα συστατικά είναι απενεργοποιημένα θα τα ενεργοποιεί και το αντίστροφο. Εννοείται πως το κουμπί *Brower* θα είναι πάντα ενεργό. Έτσι, δημιουργούμε ένα *Action* γεγονός για το κουμπί *Brower*. Με βάση αυτά που μελετήσαμε στο προηγούμενο βήμα θα δούμε τέσσερις τρόπους υλοποίησης της ενέργειας *ActionPerformed* του κουμπιού *Brower*.

Ο πρώτος τρόπος φαίνεται στην Εικόνα 16. Στη γραμμή 221 με χρήση της μεθόδου *isEnabled()* ελέγχουμε αν ένα συστατικό είναι ενεργό. Μπορούμε να κάνουμε τον έλεγχο με βάση οποιοδήποτε συστατικό εκτός του *Brower* που είναι πάντα ενεργό. Αν είναι ενεργό το συστατικό (αν ο έλεγχος *BC.isEnabled()* επιστρέψει *true*) τότε απενεργοποιούμε όλα τα συστατικά – γραμμές 222-239, αλλιώς τα ενεργοποιούμε – γραμμές 241-258.



```

219
220 private void BpowerActionPerformed(java.awt.event.ActionEvent evt) {
221     if (BC.isEnabled()) {
222         BC.setEnabled(false);
223         B0.setEnabled(false);
224         B1.setEnabled(false);
225         B2.setEnabled(false);
226         B3.setEnabled(false);
227         B4.setEnabled(false);
228         B5.setEnabled(false);
229         B6.setEnabled(false);
230         B7.setEnabled(false);
231         B8.setEnabled(false);
232         B9.setEnabled(false);
233         Bdivide.setEnabled(false);
234         Bdot.setEnabled(false);
235         Bequal.setEnabled(false);
236         Bminus.setEnabled(false);
237         Btimes.setEnabled(false);
238         Bplus.setEnabled(false);
239         calcbox.setEnabled(false);
240     } else {
241         BC.setEnabled(true);
242         B0.setEnabled(true);
243         B1.setEnabled(true);
244         B2.setEnabled(true);
245         B3.setEnabled(true);
246
247         B4.setEnabled(true);
248         B5.setEnabled(true);
249         B6.setEnabled(true);
250         B7.setEnabled(true);
251         B8.setEnabled(true);
252         B9.setEnabled(true);
253         Bdivide.setEnabled(true);
254         Bdot.setEnabled(true);
255         Bequal.setEnabled(true);
256         Bminus.setEnabled(true);
257         Btimes.setEnabled(true);
258         Bplus.setEnabled(true);
259         calcbox.setEnabled(true);
260     }
261 }
262
263 /**...3 lines */
264 public static void main(String args[]) { ...31 lines }
265

```

Εικόνα 16: Υλοποίηση της ενέργειας actionPerformed του κουμπιού Bpower – 1^{ος} τρόπος

Ο δεύτερος τρόπος φαίνεται στην Εικόνα 17. Το κουμπί Bpower υλοποιείται με τη λογική να αντιστρέφει την κατάσταση των κουμπιών. Έτσι, με την εντολή `BC.setEnabled(!BC.isEnabled())`, η παράμετρος γίνεται false όταν το `BC.isEnabled()` επιστρέφει true (με το !) και το αντίστροφο.

```

219
220 private void BpowerActionPerformed(java.awt.event.ActionEvent evt) {
221     BC.setEnabled(!BC.isEnabled());
222     B0.setEnabled(!B0.isEnabled());
223     B1.setEnabled(!B1.isEnabled());
224     B2.setEnabled(!B2.isEnabled());
225     B3.setEnabled(!B3.isEnabled());
226     B4.setEnabled(!B4.isEnabled());
227     B5.setEnabled(!B5.isEnabled());
228     B6.setEnabled(!B6.isEnabled());
229     B7.setEnabled(!B7.isEnabled());
230     B8.setEnabled(!B8.isEnabled());
231     B9.setEnabled(!B9.isEnabled());
232     Bdivide.setEnabled(!Bdivide.isEnabled());
233     Bdot.setEnabled(!Bdot.isEnabled());
234     Bequal.setEnabled(!Bequal.isEnabled());
235     Bminus.setEnabled(!Bminus.isEnabled());
236     Btimes.setEnabled(!Btimes.isEnabled());
237     Bplus.setEnabled(!Bplus.isEnabled());
238     calcbbox.setEnabled(!calcbbox.isEnabled());
239 }
240

```

Εικόνα 17: Υλοποίηση της ενέργειας ActionPerformed του κουμπιού Bpower – 2^{ος} τρόπος

Ο τρίτος τρόπος φαίνεται στην Εικόνα 18. Η προσέγγισή του βασίζεται στη δήλωση του πίνακα συστατικών *CalcComponents*, ο οποίος περιέχει όλα τα συστατικά που αλλάζουν κατάσταση.

```

219
220 private void BpowerActionPerformed(java.awt.event.ActionEvent evt) {
221     Component[] CalcComponents = {B0, B1, B2, B3, B4, B5, B6, B7, B8, B9,
222     BC, Bdivide, Bdot, Bequal, Bminus, Btimes, Bplus, calcbbox};
223     for (Component comp : CalcComponents)
224         comp.setEnabled(!comp.isEnabled());
225 }
226
227 /**...3 lines */
230 public static void main(String args[]) { ...31 lines }
261

```

Εικόνα 18: Υλοποίηση της ενέργειας ActionPerformed του κουμπιού Bpower – 3^{ος} τρόπος

Ο τέταρτος τρόπος φαίνεται στην Εικόνα 19. Η λογική της υλοποίησής του βασίζεται στη δημιουργία του πίνακα συστατικών με δυναμικό τρόπο.

```

219
220 private void BpowerActionPerformed(java.awt.event.ActionEvent evt) {
221     int compcount = getContentPane().getComponentCount();
222     for (int i = 0; i < compcount; i++) {
223         Component comp = getContentPane().getComponent(i);
224         if (comp != Bpower)
225             comp.setEnabled(!comp.isEnabled());
226     }
227 }
228
229 /**...3 lines */
232 public static void main(String args[]) { ...31 lines }
263

```

Εικόνα 19: Υλοποίηση της ενέργειας ActionPerformed του κουμπιού Bpower – 4^{ος} τρόπος

Βήμα 12: Στη συνέχεια θα προγραμματίσουμε το *Action* γεγονός των δέκα αριθμών κουμπιών ώστε όταν τα επιλέγει ο χρήστης να γράφεται ένας αριθμός δεξιά των αριθμών που έχουν γραφτεί στο *calcbox*. Έτσι, αφού δημιουργήσουμε το *Action* γεγονός ενός κουμπιού υλοποιούμε την ενέργεια *ActionPerformed*, όπως φαίνεται στην Εικόνα 20. Για να τοποθετηθεί ο αριθμός που επιλέγουμε δεξιά αυτών που υπάρχουν ήδη στο *calcbox* θα πρέπει πρώτα να διαβάσουμε το περιεχόμενο του *calcbox* με τη μέθοδο *getText()* και στη συνέχεια να προσθέσουμε (με string concatenation) τον αριθμό.



```

279 private void B0ActionPerformed(java.awt.event.ActionEvent evt) {
280     calcbox.setText(calcbox.getText()+"0");
281 }
282
283 private void B1ActionPerformed(java.awt.event.ActionEvent evt) {
284     calcbox.setText(calcbox.getText()+"1");
285 }
286
287 private void B2ActionPerformed(java.awt.event.ActionEvent evt) {
288     calcbox.setText(calcbox.getText()+"2");
289 }
290
291 private void B3ActionPerformed(java.awt.event.ActionEvent evt) {
292     calcbox.setText(calcbox.getText()+"3");
293 }
294
295 private void B4ActionPerformed(java.awt.event.ActionEvent evt) {
296     calcbox.setText(calcbox.getText()+"4");
297 }
298
299 private void B5ActionPerformed(java.awt.event.ActionEvent evt) {
300     calcbox.setText(calcbox.getText()+"5");
301 }
302
303 private void B6ActionPerformed(java.awt.event.ActionEvent evt) {
304     calcbox.setText(calcbox.getText()+"6");
305 }
306
307 private void B7ActionPerformed(java.awt.event.ActionEvent evt) {
308     calcbox.setText(calcbox.getText()+"7");
309 }
310
311 private void B8ActionPerformed(java.awt.event.ActionEvent evt) {
312     calcbox.setText(calcbox.getText()+"8");
313 }
314
315 private void B9ActionPerformed(java.awt.event.ActionEvent evt) {
316     calcbox.setText(calcbox.getText()+"9");
317 }
318

```

Εικόνα 20: Υλοποίηση των γεγονότων *Action* των κουμπιών των δέκα αριθμών

Βήμα 13: Η υλοποίηση του κουμπιού της τελείας μπορεί να γίνει με δύο τρόπους. Καταρχήν, την τελεία θα την γράψουμε στο *calcbox* όπως όλους τους αριθμούς. Φυσικά και στους δύο τρόπους θα πρέπει να εξασφαλίσουμε ότι η τελεία θα γράφεται μόνο μία φορά στον αριθμό.

Στην Εικόνα 21, παρουσιάζεται ο πρώτος τρόπος όπου χρησιμοποιείται μία *σημαία* για να ελέγχει κάθε φορά αν περιέχει τελεία ο αριθμός που είναι στο *calcbox*. Έτσι, στη γραμμή 366, έξω από μεθόδους (και την *main*) μέσα στην τάξη, δηλώνουμε ως *global* τη Boolean μεταβλητή *dotflag*, η οποία θα έχει το ρόλο της σημαίας. Αρχική τιμή της μεταβλητής ορίζουμε

την τιμή *false* που σημαίνει ότι δεν έχει μπει τελεία. Έτσι, στον κώδικα της ενέργειας *ActionPerformed* για την τελεία, ελέγχεται πρώτα η τιμή της *dotflag*. Αν είναι *false*, άρα δεν έχει μπει τελεία, τότε προσθέτουμε μία τελεία στο *calcbbox*, δεξιά του υπάρχοντος αριθμού και αλλάζουμε την τιμή της *dotflag* σε *true*. Τέλος, κάθε φορά που γράφεται νέος αριθμός, αφού αδειάσει το *calcbbox*, θα πρέπει η *dotflag* να γίνεται *false*.

```

323
324 private void BdotActionPerformed(java.awt.event.ActionEvent evt) {
325     if (dotflag == false) {
326         calcbbox.setText(calcbbox.getText() + ".");
327         dotflag = true;
328     }
329 }
330
331 /**...3 lines */
334 public static void main(String args[]) {...31 lines }
365
366     boolean dotflag = false;
367
368     // Variables declaration - do not modify
369     private javax.swing.JButton B0;
370     private javax.swing.JButton B1;

```

Εικόνα 21: Υλοποίηση του γεγονότος Action του κουμπιού της τελείας – 1^{ος} τρόπος

Στην Εικόνα 22, παρουσιάζεται ο δεύτερος τρόπος όπου αφού γραφτεί η τελεία στο *calcbbox* απενεργοποιείται το κουμπί της τελείας για να μην μπορεί ο Χρήστης να ξαναγράψει την τελεία. Φυσικά, όταν θα αδειάζει το *calcbbox* θα πρέπει να ενεργοποιείται το κουμπί της τελείας.

```

323
324 private void BdotActionPerformed(java.awt.event.ActionEvent evt) {
325     calcbbox.setText(calcbbox.getText() + ".");
326     Bdot.setEnabled(false);
327 }
328

```

Εικόνα 22: Υλοποίηση του γεγονότος Action του κουμπιού της τελείας – 2^{ος} τρόπος

Βήμα 14: Το αριθμητήριο που παρουσιάζουμε είναι αρκετά απλό αφού θα διαχειρίζεται την πράξη δύο αριθμών. Στην Εικόνα 23, παρουσιάζεται ο κώδικας των τεσσάρων πράξεων. Έτσι, στα Action γεγονότα των κουμπιών των τεσσάρων πράξεων θα πρέπει να γίνουν τα παρακάτω:

- Να καταχωρηθεί ο πρώτος αριθμός της πράξης, ο αριθμός που είχε καταχωρηθεί στο *calcbbox* πριν πατηθεί το πλήκτρο της πράξης σε μία *double* μεταβλητή – γραμμή 350. Η μεταβλητή *number* θα πρέπει να δηλωθεί ως *global*.
- Να ετοιμαστεί το *calcbbox* να δεχτεί τον δεύτερο αριθμό. Με άλλα λόγια να αδειάσει το περιεχόμενό του – γραμμή 351.
- Το κουμπί της τελείας να αρχικοποιηθεί – γραμμή 352. Σε περίπτωση που προτιμήθηκε η λύση με τη σημαία θα πρέπει να χρησιμοποιηθεί ο κώδικας στο σχόλιο της γραμμής 352.
- Να σημειωθεί σε μία *global String* μεταβλητή το σύμβολο της πράξης. Για παράδειγμα αν επιλεγεί η πράξη της πρόσθεσης θα σημειωθεί η πρόσθεση – γραμμή 353.

```

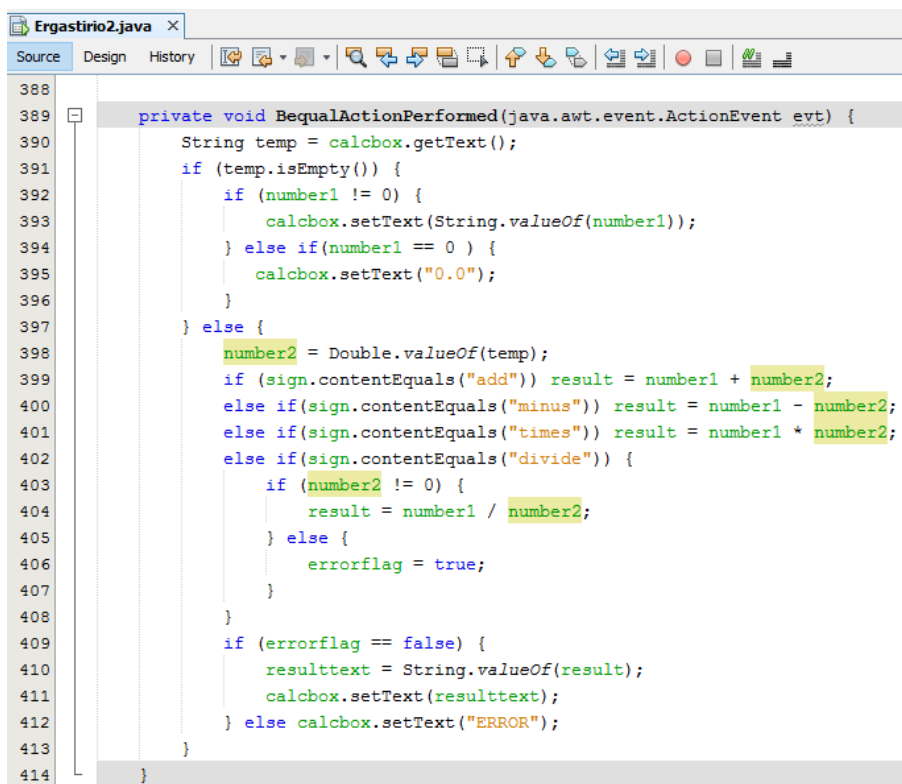
349 private void BplusActionPerformed(java.awt.event.ActionEvent evt) {
350     number1 = Double.valueOf(calcbox.getText());
351     calcbox.setText("");
352     Bdot.setEnabled(true);    // εναλλακτικά dotflag = false;
353     sign = "add";
354 }
355
356 private void BminusActionPerformed(java.awt.event.ActionEvent evt) {
357     number1 = Double.valueOf(calcbox.getText());
358     calcbox.setText("");
359     Bdot.setEnabled(true);    // εναλλακτικά dotflag = false;
360     sign = "minus";
361 }
362
363 private void BtimesActionPerformed(java.awt.event.ActionEvent evt) {
364     number1 = Double.valueOf(calcbox.getText());
365     calcbox.setText("");
366     Bdot.setEnabled(true);    // εναλλακτικά dotflag = false;
367     sign = "times";
368 }
369
370 private void BdivideActionPerformed(java.awt.event.ActionEvent evt) {
371     number1 = Double.valueOf(calcbox.getText());
372     calcbox.setText("");
373     Bdot.setEnabled(true);    // εναλλακτικά dotflag = false;
374     sign = "divide";
375 }
376

```

Εικόνα 23: Ο κώδικας των τεσσάρων πράξεων

Βήμα 15: Σε αυτό το βήμα θα προγραμματιστεί το ίσον. Έτσι, στην ενέργεια *ActionPerformed* του κουμπιού *Bequal* (Εικόνα 24) θα πρέπει να γίνουν τα παρακάτω:

- Γραμμή 390, αποθηκεύουμε σε μία String temp μεταβλητή το περιεχόμενο του calcbox, που λογικά θα περιέχει το δεύτερο αριθμό.
- Γραμμές 391 – 396, αν ο χρήστης πάτησε το ίσον πριν βάλει δεύτερο αριθμό τότε το αποτέλεσμα θα είναι ο πρώτος αριθμός, αν υπάρχει, αλλιώς το μηδέν.
- Αν ο χρήστης έβαλε δεύτερο αριθμό πριν πατήσει το ίσον, τότε στη γραμμή 398, μετατρέπεται σε double και αποθηκεύεται στη μεταβλητή *number2*.
- Γραμμή 399, αν η πράξη είναι πρόσθεση τότε γίνεται πρόσθεση.
- Γραμμή 400, αν η πράξη είναι αφαίρεση τότε γίνεται αφαίρεση.
- Γραμμή 401, αν η πράξη είναι πολλαπλασιασμός τότε γίνεται πολλαπλασιασμός.
- Γραμμές 402-408, αν η πράξη είναι διαίρεση τότε γίνεται διαίρεση και έλεγχος αν ο διαιρέτης είναι το μηδέν.
- Γραμμές 409-413, εμφανίζεται το αποτέλεσμα.



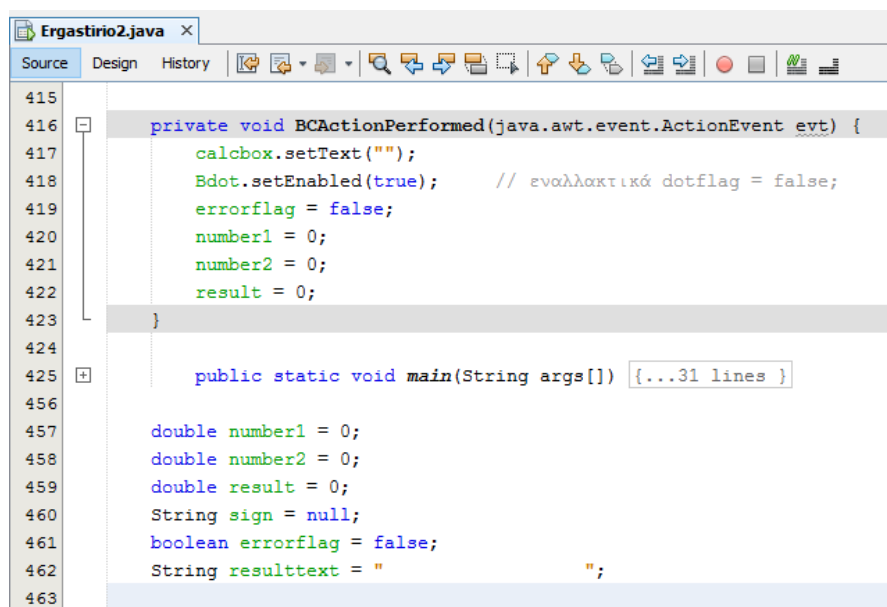
```

388
389 private void BequalActionPerformed(java.awt.event.ActionEvent evt) {
390     String temp = calcbox.getText();
391     if (temp.isEmpty()) {
392         if (number1 != 0) {
393             calcbox.setText(String.valueOf(number1));
394         } else if (number1 == 0) {
395             calcbox.setText("0.0");
396         }
397     } else {
398         number2 = Double.valueOf(temp);
399         if (sign.contentEquals("add")) result = number1 + number2;
400         else if (sign.contentEquals("minus")) result = number1 - number2;
401         else if (sign.contentEquals("times")) result = number1 * number2;
402         else if (sign.contentEquals("divide")) {
403             if (number2 != 0) {
404                 result = number1 / number2;
405             } else {
406                 errorflag = true;
407             }
408         }
409         if (errorflag == false) {
410             resulttext = String.valueOf(result);
411             calcbox.setText(resulttext);
412         } else calcbox.setText("ERROR");
413     }
414 }

```

Εικόνα 24: Η ενέργεια ActionPerformed του ίσον (Bequal)

Βήμα 16: Για να αδειάσουμε το *textfield* *calcbox* θα χρησιμοποιήσουμε το κουμπί C. Με αυτήν την ενέργεια ξεκινάμε μια νέα πράξη. Οπότε είναι σημαντικό να αρχικοποιήσουμε όλες τις παραμέτρους του αριθμητήριου μας. Ο σχετικός κώδικας παρουσιάζεται στην Εικόνα 25, καθώς και η δήλωση όλων των global μεταβλητών (γραμμές 457-462).



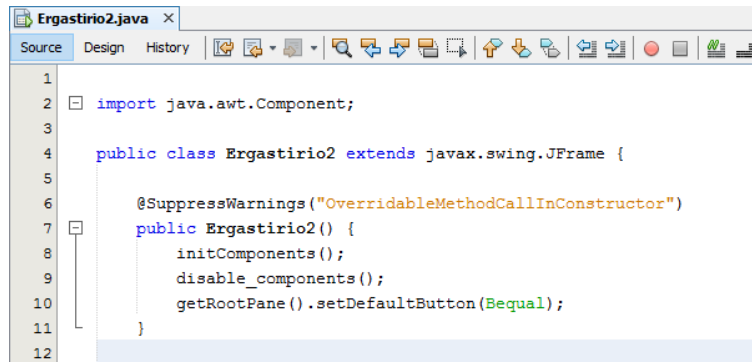
```

415
416 private void BCActionPerformed(java.awt.event.ActionEvent evt) {
417     calcbox.setText("");
418     Bdot.setEnabled(true); // εναλλακτικά dotflag = false;
419     errorflag = false;
420     number1 = 0;
421     number2 = 0;
422     result = 0;
423 }
424
425 public static void main(String args[]) { ...31 lines }
426
427 double number1 = 0;
428 double number2 = 0;
429 double result = 0;
430 String sign = null;
431 boolean errorflag = false;
432 String resulttext = "";
433

```

Εικόνα 25: Η ενέργεια ActionPerformed του κουμπιού που καθαρίζει το textfield (BC)

Βήμα 17: Τελευταίο βήμα είναι ο προγραμματισμός του <ENTER> να κάνει την ίδια ενέργεια με το ίσον (το κουμπί *Bequal*), όποτε το επιλέγει/πατάει ο χρήστης, ενώ ο δείκτης του ποντικιού είναι μέσα στο *textfield calcbbox*. Στη γραμμή 10 της Εικόνα 26 παρουσιάζεται ο κώδικας που υλοποιεί το ζητούμενο. Το *defaultButton* είναι το <ENTER>, το οποίο ορίζεται, όποτε επιλέγεται, με την *setDefaultButton* μέθοδο να εκτελεί την ενέργεια που έχει προγραμματιστεί για το *Bequal*.



```
1
2 import java.awt.Component;
3
4 public class Ergastirio2 extends javax.swing.JFrame {
5
6     @SuppressWarnings("OverrideMethodCallInConstructor")
7     public Ergastirio2() {
8         initComponents();
9         disable_components();
10        getRootPane().setDefaultButton(Bequal);
11    }
12 }
```

Εικόνα 26: Το <ENTER> λειτουργεί ως ίσον