Κέντρο ΠΛΗ.ΝΕ.Τ. Ν. Φλώρινας

΄Ετοιμα Προγράμματα σε Java

Σειρά 1η

Φλώρινα, Μάρτιος 2008

Πρόγραμμα 1 – Ένα Πολύ Απλό Πρόγραμμα σε Java

```
class Program1 {
        public static void main(String[ ] args) {
                System.out.println("Hello from Java");
        }
}
```

Όλα τα προγράμματα της Java αποτελούν μια *τάξη ή κλάση (class)* και το όνομά τους πρέπει να αρχίζει με κεφαλαίο γράμμα. Κάθε πρόγραμμα σε Java πρέπει να περιέχει μια μέθοδο με όνομα *main()*, η οποία καθορίζει το σημείο απ' όπου αρχίζει η εκτέλεση της εφαρμογής. Το *public* κάνει την εφαρμογή προσβάσιμη απ' άλλες εφαρμογές.

Η εντολή (μέθοδος ή συνάρτηση) *println* αποτελεί τμήμα της ενότητας *out* του αντικειμένου *System* και χρησιμοποιείται για να εμφανίσει ένα μήνυμα στην οθόνη. Το επίθεμα ln (Line Feed) προκαλεί αλλαγή γραμμής μετά την εμφάνιση του μηνύματος. Κάθε εντολή της Java τελειώνει μ' ένα ;.

Τα προγράμματα της Java τα γράφουμε σαν αρχεία κειμένου με επέκταση *.java* και το όνομα του αρχείου πρέπει να είναι το ίδιο με το όνομα της τάξης.

Για να μεταγλωττίσουμε ένα πρόγραμμα της Java, πάμε στη γραμμή εντολών του Ms-Dos και γράφουμε *javac Program1.java*, οπότε θα παραχθεί ένα αρχείο με το όνομα *Program1.class*. Αν ο μεταγλωττιστής εντοπίσει λάθη, θα μας ενημερώσει για το ποια είναι και σε ποια γραμμή βρίσκονται. Για να εκτελέσουμε την εφαρμογή, πρέπει να γράψουμε *java Program1*.

Πρόγραμμα 2 – Το Πρώτο μας Applet

Τα applets είναι προγράμματα της Java που φιλοξενούνται και εκτελούνται μέσα από έναν φυλλομετρητή (browser) και ένα απλό πρόγραμμα applet είναι το παρακάτω :

```
import java.applet.Applet;
import java.awt.Graphics;

public class Program2 extends Applet {
        public void paint(Graphics g) {
                g.drawString("Hello from an applet!", 50, 20);
        }
}
```

Τα applets είναι αντικείμενα που ανήκουν στην τάξη Applet, που είναι μια από τις τάξεις της ομάδας τάξεων (πακέτου) java.applet και όλες οι εμφανίσεις στην οθόνη γίνονται μέσω αντικειμένων της τάξης Graphics του συνόλου java.awt (Abstract Windows Toolkit).

Για να μπορέσουμε να χρησιμοποιήσουμε αυτές τις τάξεις θα πρέπει να ενσωματώσουμε τον κώδικά τους μέσα στο πρόγραμμα με την εντολή *import*. Θα χρειαζόμαστε πάντα την τάξη java. applet.Applet και συνήθως μια ή παραπάνω τάξεις από το σύνολο java.awt.

Η *drawString* είναι μια από τις μεθόδους (συναρτήσεις) που εμφανίζουν κείμενο στην οθόνη και χρησιμοποιεί τρεις παραμέτρους : το κείμενο και τις συντεταγμένες (x και y) της οθόνης, όπου θα εμφανισθεί το κείμενο.

Εκτελούμε ένα applet μέσα από ένα έγγραφο HTML, όπως ο παρακάτω κώδικας HTML που καλεί την εφαρμογή applet με όνομα Program2 και την εμφανίζει σ' ένα πλαίσιο με πλάτος 200 pixels και ύψος 50 pixels.

Ένα applet μπορούμε να το δούμε να εκτελείται και από τη γραμμή εντολής αν γράψουμε *appletviewer Program2.html*, οπότε θα εμφανισθεί ένα παράθυρο μ' ένα μενού στα αριστερά του.

```
<html>
<head>
        <title>Το Πρώτο μας Applet</title>
</head>
<body>
        <br>
        <applet code="Program2.class" width=200 height=50>
        </applet>
</body>
</html>
```

Πρόγραμμα 3 – Η Μέθοδος Print

Με τη μέθοδο *print()*, ή την *println()* για αλλαγή γραμμής, μπορούμε να εμφανίσουμε σταθερές τιμές και μεταβλητές στην οθόνη και να τις ενώσουμε με τον τελεστή +, ως εξής :

```
class Program3 {
        public static void main(String[ ] args) {
                // ορισμός των μεταβλητών
                int number = 100;
                long bigNum = 2000000;
                char letter = 'A';
                boolean reply = true;

                // εμφάνιση των τιμών των μεταβλητών
                System.out.println("Ο ακέραιος είναι : " + number);
                System.out.println("Ο μεγάλος ακέραιος : " + bigNum);
                System.out.println("Ο χαρακτήρας είναι : " + letter);
                System.out.println("Η λογική μεταβλητή είναι : " + reply);
        }
}
```

Πρόγραμμα 4 – Η Μέθοδος Read

Για να διαβάσουμε μια τιμή από το πληκτρολόγιο, χρησιμοποιούμε τη μέθοδο *read()*, ως εξής :

```
a = System.in.read();
```

Η μέθοδος read() επιστρέφει μια ακέραια τιμή τύπου int και όχι έναν χαρακτήρα (char), αλλά στη Java μπορούμε πολύ εύκολα να μετατρέπουμε δεδομένα από τη μια μορφή στην άλλη με τη χρήση μετατροπών τύπων, γράφοντας τον επιθυμητό τύπο μέσα σε παρενθέσεις μπροστά από τα δεδομένα, όπως στα παρακάτω παραδείγματα :

```
letter = (char) input_data;
c = (char) System.in.read();
```

Ακολουθεί ένα πρόγραμμα σε Java που χρησιμοποιεί τη μέθοδο read().

```
class Program4 {
        public static void main(String[ ] args) {
                char c;

                System.out.println("Πατήστε ένα πλήκτρο και enter");

                // διάβασμα του χαρακτήρα και μετατροπή του σε char
                c = (char) System.in.read();

                // εμφάνιση του χαρακτήρα
                System.out.println("Πατήσατε το : " + c);
        }
}
```

Πρόγραμμα 5 – Οι Πίνακες (Arrays) στην Java

Στους πίνακες (arrays) της Java, ο δείκτης γράφεται μέσα στις γνωστές αγκύλες [ ] μετά το όνομα του πίνακα και μια τυπική δήλωση για έναν πίνακα ακεραίων τιμών 100 θέσεων είναι η εξής :

```
int[ ] number = new int[100];
```

Με την παραπάνω εντολή ορίζουμε έναν πίνακα ακεραίων (int) με το όνομα number και με αρίθμηση από το 0 έως το 99.

Ακολουθεί ένα απλό πρόγραμμα με χρήση πίνακα ακεραίων.

```
class Program5 {
        public static void main(String[ ] args) {
                // ορίζουμε έναν πίνακα ακεραίων (int) με όνομα
                // number και με αρίθμηση από το 0 έως το 4
                int[ ] number = new int[5];
                int i;

                number[0] = 15;
                number[1] = 25;
                number[2] = 30;
                number[3] = 40;
                number[4] = 45;

                // εμφανίζουμε την τιμή του 1ου [0] στοιχείου του πίνακα
                System.out.println(number[0]);

                // εμφανίζουμε την τιμή του 4ου [3] στοιχείου του πίνακα
                i = 3;
                System.out.println(number[i]);
        }
}
```

Πρόγραμμα 6 – Τα Strings στην Java

Η Java χρησιμοποιεί την τάξη *string* για την αποθήκευση κειμένου που είναι σταθερό και που συνήθως χρησιμοποιείται για την αποθήκευση μηνυμάτων και για τη μεταφορά δεδομένων ανάμεσα σε μεθόδους.

```
class Program6 {
        public static void main(String[ ] args) {
                // ορισμός ενός νέου string με όνομα message
                String message = new String();
                // θα μπορούσαμε να γράψουμε και απλά String message;

                // εκχώρηση κειμένου στο string message
                message = "Florina per sempre";
                // εμφάνιση του περιεχομένου του string message
                System.out.println(message);

                // ανάθεση νέας τιμής στο string message
                message = "Το string άλλαξε τιμή";
                // εμφάνιση εκ νέου του περιεχομένου του string message
                System.out.println(message);
        }
}
```

Πρόγραμμα 7 – Οι Παράμετροι (Arguments)

Η έκφραση main(String[ ] args) δηλώνει έναν πίνακα από String, ο οποίος πίνακας θα συλλέξει οποιοδήποτε κείμενο πληκτρολογηθεί από τον χρήστη στη γραμμή εντολών μετά το όνομα του προγράμματος.

Όταν εκτελεσθεί το πρόγραμμα, το σύστημα θα αντιγράψει το πρώτο στοιχείο που βρίσκεται μετά το όνομα του προγράμματος στο args[0], το δεύτερο στοιχείο στο args[1] κοκ. Έτσι, μπορούμε να δώσουμε μαζί με την κλήση ενός προγράμματος και ορισμένες παραμέτρους.

```
class Program7 {
        public static void main(String[ ] args) {
                // δήλωση των δύο String μεταβλητών
                String firstName;
                String surName;

                // εκχώρηση των τιμών των δύο παραμέτρων
                // στις δύο String μεταβλητές
                firstName = args[0];
                surName = args[1];
                // εμφάνιση των τιμών των δύο παραμέτρων
                System.out.println("Γεια "+firstName+" " + surName);
        }
}
```

Το παραπάνω πρόγραμμα αντιγράφει τις τιμές του πίνακα args[] στις συμβολοσειρές firstName και surName και στη συνέχεια χαιρετά τον χρήστη. Όταν θα εκτελέσουμε το πρόγραμμα, πρέπει να γράψουμε δίπλα στο όνομα του προγράμματος στη γραμμή εντολών ένα όνομα και ένα επώνυμο. Αν γράψουμε λιγότερα ονόματα, το πρόγραμμα θα σταματήσει, ενώ αν γράψουμε περισσότερα ονόματα, τα παραπάνω θα αγνοηθούν.

Πρόγραμμα 8 – Τα StringBuffers στην Java – 1ο

Τα δεδομένα που περιέχονται στα αντικείμενα StringBuffers μπορούν να υποστούν επεξεργασία από το πρόγραμμα και έτσι αν θέλουμε να κάνουμε μια επεξεργασία σ' ένα κείμενο, όπως προσθήκη ή αφαίρεση κειμένου, πρέπει να το τοποθετήσουμε σ' ένα StringBuffer.

Τα αντικείμενα StringBuffer δημιουργούνται με τρεις τρόπους :

```
// δημιουργία ενός StringBuffer με αρχικό μέγεθος χαρακτήρων 16
// (default)
StringBuffer s1 = new StringBuffer();
// δημιουργία ενός StringBuffer με αρχικό μέγεθος χαρακτήρων 40
StringBuffer s2 = new StringBuffer(40);
// δημιουργία ενός StringBuffer με αρχική τιμή και με
// μέγεθος χαρακτήρων 4
StringBuffer s3 = new StringBuffer("Java");
```

Δεν μπορούμε να δώσουμε τιμή σ' ένα αντικείμενο StringBuffer με το σύμβολο = και οι τιμές πρέπει να δίνονται κατά τη δημιουργία του αντικειμένου. Με τη μέθοδο *append()* μπορούμε να προσθέσουμε κείμενο στο τέλος και με τη μέθοδο *insert()* μπορούμε να παρεμβάλουμε κείμενο σε κάποιο σημείο μιας συμβολοσειράς.

```
class Program8 {
        public static void main(String[ ] args) {
                // ανάθεση αρχικής τιμής στη συμβολοσειρά
                StringBuffer s1 = new StringBuffer("Java ");
                System.out.println(s1);

                // προσθήκη κειμένου στο τέλος της συμβολοσειράς
                s1.append("Γλώσσα Προγραμματισμού");
                System.out.println(s1);

                // προσθήκη κειμένου στη μέση της συμβολοσειράς
                // και στη θέση 5
                s1.insert(5, "Μια ");
                System.out.println(s1);
        }
}
```

Πρόγραμμα 9 – Τα StringBuffers στην Java – 2ο

Στο παρακάτω πρόγραμμα διαβάζουμε συνέχεια χαρακτήρες από το πληκτρολόγιο, τους προσθέτουμε σε μια συμβολοσειρά τύπου StringBuffer και τελειώνουμε όταν πατηθεί το πλήκτρο <enter> αντί για όνομα.

```
class Program9 {
        public static void main(String[ ] args) {
                char c;
                StringBuffer name = new StringBuffer();
                // το αρχικό StringBuffer είναι άδειο

                System.out.println("Γράψτε ένα όνομα και <enter>");
                do {
                        // ανάγνωση ενός χαρακτήρα και
                        // προσθήκη του στο όνομα
                        c = (char) System.in.read();
                        name.append(c);
                } while (c != '\n');
                System.out.println("Γεια σου "+name);
        }
}
```

Πρόγραμμα 10 – Τοπικές και Καθολικές Μεταβλητές

Οι μεταβλητές που δηλώνονται στο επίπεδο της τάξης (καθολικές μεταβλητές), δηλ. στην αρχή του προγράμματος, μπορούν να χρησιμοποιηθούν σ' οποιοδήποτε σημείο του προγράμματος. Οι δηλώσεις αυτές πρέπει να ξεκινούν με τη λέξη *static*, που δεσμεύει μόνιμα χώρο μνήμης για τη μεταβλητή.

Οι μεταβλητές που δηλώνονται μέσα σε μια μέθοδο (τοπικές μεταβλητές) μπορούν να χρησιμοποιηθούν μόνο μέσα στη μέθοδο. Μεταβλητές μπορούν να δηλωθούν και μέσα σε δομές, όπως σε βρόχους for, υπάρχουν μόνο στα πλαίσια του βρόχου και εξαφανίζονται όταν βγούμε απ' αυτούς.

Το παρακάτω πρόγραμμα περιέχει δύο μεθόδους και χρησιμοποιεί την καθολική μεταβλητή b, που αναγνωρίζεται και προσπελάζεται και από τις δύο μεθόδους, και δύο μεταβλητές με το όνομα a που είναι τοπικές σε κάθε μέθοδο και που δεν έχουν καμία σχέση μεταξύ τους.

```
class Program10 {
        // δήλωση καθολικής μεταβλητής σε επίπεδο τάξης
        static int b = 0;

        public static void main(String[ ] args) {
                int a = 10;     // δήλωση τοπικής μεταβλητής

                b = 20;
                // αλλαγή της τιμής της καθολικής μεταβλητής b
                minor();        // κλήση της μεθόδου minor()
        }

        public static void minor() {
                int a = -1;     // δήλωση τοπικής μεταβλητής

                b = 21;
                // αλλαγή της τιμής της καθολικής μεταβλητής b
        }
}
```

Πρόγραμμα 11 – Ο Βρόχος For

Το παρακάτω πρόγραμμα εκτυπώνει την προπαίδεια του 7, από το 1 έως και το 10, με τη βοήθεια του βρόχου for.

```
class Program11 {
        public static void main(String[ ] args) {
                int loop;               // η μεταβλητή του βρόχου
                int table = 7;          // ο αριθμός της προπαίδειας
                for (loop = 1; loop <= 10; loop++)
                        System.out.println(loop * table);
        }
}
```

Πρόγραμμα 12 – Ο Διπλός Βρόχος For

Το παρακάτω πρόγραμμα εκτυπώνει τις προπαίδειες των αριθμών από το 1 έως το 6 και χρησιμοποιεί δύο βρόχους for, τον έναν μέσα στον άλλον.

```
class Program11 {
        public static void main(String[ ] args) {
                int outer, inner;

                for (outer = 1; outer <= 6; outer++) {
                        for (inner = 1; inner <= 10; inner++)
                                System.out.println(inner * outer);
                        System.out.println(); // αλλαγή γραμμής
                } // τέλος του εξωτερικού for
        }
}
```

Πρόγραμμα 13 – Ο Βρόχος While

Στον βρόχο while ο έλεγχος γίνεται κατά την είσοδο στον βρόχο και πριν από κάθε επανάληψη.

```
class Program13 {
        public static void main(String[ ] args) {
                int count = 0;

                // θα εκτυπωθούν οι αριθμοί από το 1 έως και το 10
                while (count < 10) {
                        count ++;
                        // αύξηση κατά ένα πριν από την εκτύπωση
                        System.out.println(count);
                } // τέλος του while
        }
}
```

Πρόγραμμα 14 – Ο Βρόχος Do ... While

Στον βρόχο do ... while ο έλεγχος γίνεται στο τέλος του βρόχου και έτσι οι εντολές που βρίσκονται μέσα στο σώμα του βρόχου εκτελούνται τουλάχιστον μία φορά.

Το παρακάτω πρόγραμμα χρησιμοποιεί τον βρόχο do ... while και δημιουργεί συνεχώς τυχαίους αριθμούς από το 0 έως και το 9, τελειώνει δε όταν παραχθεί ο αριθμός 6.

```
class Program14 {
        public static void main(String[ ] args) {
                int num = 0;

                System.out.println("Αναζήτηση του αριθμού 6");
                do {
                        // δημιουργία ενός τυχαίου ακεραίου αριθμού
                        // από το 0 έως το 9
                        num = (int) (java.lang.Math.random() * 10);
                        System.out.print(num +" ");
                } while (num != 6);
                System.out.println("Το 6 βρέθηκε!");
        }
}
```

Στην τάξη *java.lang.Math* βρίσκεται η μέθοδος *random()*, με την οποία μπορούμε να δημιουργήσουμε έναν τυχαίο αριθμό από το 0,0 έως το 0,9. Η χρήση της γίνεται ως εξής :

```
x = java.lang.Math.random();
```

Η παραπάνω εντολή επιστρέφει μια πραγματική θετική τιμή μικρότερη από το 1 και αν θέλουμε ακέραιους αριθμούς μέχρι ένα όριο, π.χ. 100, θα πρέπει να χρησιμοποιήσουμε το (int) στην αρχή και να πολλαπλασιάσουμε με το 100, ως εξής :

```
randNum = (int) (java.lang.Math.random() * 100);
```

Το αποτέλεσμα θα είναι ένας τυχαίος ακέραιος αριθμός από το 0 έως το 99.

Πρόγραμμα 15 – Η Εντολή Switch

Μια πολύ συνηθισμένη χρήση της εντολής Switch είναι για τη δημιουργία ενός μενού, όπως στο παρακάτω πρόγραμμα.

```
class Program15 {
        public static void main(String[ ] args) {
                char choice;

                System.out.println("Δημιουργία αρχείου....1");
                System.out.println("Άνοιγμα αρχείου.....2");
                System.out.println("Αποθήκευση αρχείου...3");
                System.out.println("Έξοδος.....4");

          do {
                System.out.println("Επιλέξτε από 1-4 και <enter> : ");
                choice = (char) System.in.read();
                switch (choice) {
                        case '1' : System.out.println("Εκκίνηση"); break;
                        case '2' : System.out.println("Άνοιγμα"); break;
                        case '3' : System.out.println("Αποθήκευση"); break;
                        case '4' : System.out.println("Αντίο"); break;
                        default : System.out.println("Μη αποδεκτή");
                } // τέλος του switch
          } while (choice != '4');
        }
}
```

Πρόγραμμα 16 – Οι Μέθοδοι

Οι γνωστές από άλλες γλώσσες προγραμματισμού διαδικασίες και συναρτήσεις αποκαλούνται μέθοδοι στην Java και στο παρακάτω πρόγραμμα έχουμε τη βασική και απαραίτητη μέθοδο κάθε προγράμματος σε Java, που είναι η main(), και τη μέθοδο powerOf(), η οποία υπολογίζει τη δύναμη ενός ακέραιου αριθμού και καλείται από τη μέθοδο main().

```
class Program16 {
        public static void main(String[ ] args) {
                int number = 4;
                int p = 3;
                // θα υπολογίσουμε το 4 εις την 3η

                // κλήση της μεθόδου για τον υπολογισμό της δύναμης
                powerOf(number, p);
                // οι τιμές των number και p μεταφέρονται στις
                // μεταβλητές n και p της μεθόδου
        }
```

```
        public static void powerOf(int n, int p) {
                // δήλωση τοπικών μεταβλητών της μεθόδου
                int temp = 1;
                int loop;

                for (loop = 1; loop <= p; loop++)
                        temp = temp * n;
                System.out.println(n+" υψωμένο στη "+p+" = "+temp);
        }
}
```

Στη γραμμή δήλωσης της μεθόδου, το *public* λέει ότι η μέθοδος μπορεί να προσπελαστεί από άλλες μεθόδους, το *static* δεσμεύει μόνιμο χώρο στη μνήμη για τις μεταβλητές της μεθόδου και το *void* διευκρινίζει ότι η μέθοδος δεν επιστρέφει καμία τιμή στο όνομά της.

Πρόγραμμα 17 – Μέθοδος που Επιστρέφει Τιμή

Ακολουθεί ξανά το προηγούμενο πρόγραμμα, αλλά αυτή τη φορά με τη μέθοδο powerOf() να επιστρέφει μια ακέραια τιμή στο όνομά της με χρήση της εντολής return, οπότε δεν χρησιμοποιούμε τη δήλωση void.

```
class Program17 {
        public static void main(String[ ] args) {
                int number = 4;
                int p = 3, ans;
                // θα υπολογίσουμε το 4 εις την 3η

                // κλήση της μεθόδου για τον υπολογισμό της δύναμης
                // και εκχώρηση του αποτελέσματος στη μεταβλητή ans
                ans = powerOf(number, p);
                System.out.println(number+" υψωμένο  "+p+" = "+ans);
        }

        public static int powerOf(int n, int p) {
                // δήλωση τοπικών μεταβλητών της μεθόδου
                int temp = 1;
                int loop;

                for (loop = 1; loop <= p; loop++)
                        temp = temp * n;
                return temp;
                // τιμή επιστροφής στην καλούσα μέθοδο
        }
}
```

Οι μέθοδοι void συμπεριφέρονται σαν διαδικασίες (procedures), ενώ οι μέθοδοι που επιστρέφουν μια τιμή, π.χ. int, συμπεριφέρονται σαν συναρτήσεις (functions).

Πρόγραμμα 18 – Ένα Απλό Πρόγραμμα Applet

Ακολουθεί ένα απλό πρόγραμμα applet για να μάθουμε τις βασικές συναρτήσεις. Στο παρακάτω παράδειγμα applet οι μέθοδοι init(), start() και stop() περιέχουν μια εντολή που προσθέτει κείμενο σ' ένα StringBuffer με το όνομα message, ενώ η μέθοδος paint() εμφανίζει το συνολικό κείμενο.

Αποθηκεύουμε το applet με το όνομα Program18.java, το μεταγλωττίζουμε κανονικά και το συνδέουμε μ' ένα αρχείο HTML με τα tags <applet> και </applet>.

```
// προσθήκη των απαραίτητων τάξεων μέσω πακέτων
import java.awt.*;
import java.applet.*;

public class Program18 extends Applet {
        StringBuffer message;

        public void init() {
                // αρχικό κείμενο του μηνύματος
                message = new StringBuffer("Florina per sempre ...");
        }

        public void start() {
                // προσθήκη στο μήνυμα κατά την έναρξη του applet
                message.append("Started ...");
        }

        public void stop() {
                // προσθήκη στο μήνυμα κατά τον τερματισμό του applet
                // το κείμενο αυτό δεν θα φανεί
                message.append("Stopped ...");
        }

        public void paint(Graphics g) {
                // εμφάνιση του ολοκληρωμένου μηνύματος
                // Florina per sempre ... Started ...
                g.drawString(message.toString(), 150, 50);
        }
}
```

Δεν πρέπει να ξεχνάμε ότι τα applets εκτελούν αυτόματα τις μεθόδους *init()* και *start()* όποτε ανοίγει η ιστοσελίδα που τα φιλοξενεί, μόνο τη μέθοδο *start()* όποτε ξανανοίγει η ιστοσελίδα αυτή και τη μέθοδο *stop()* όποτε το πρόγραμμα πλοήγησης πάει σε κάποια άλλη ιστοσελίδα.

Ο κώδικας της ιστοσελίδας που θα καλεί το παραπάνω applet είναι ο εξής :

```
<html>
<body>
        <applet code="Program18.class" width=600 height=200>
        </applet>
</body>
</html>
```

Πρόγραμμα 19 – Εμφάνιση Κειμένου σε Applet

Το παρακάτω πρόγραμμα τοποθετεί διάφορα κείμενα στις τέσσερις γωνίες και στο κέντρο ενός παραθύρου applet.

```
import java.awt.*;
import java.applet.*;

public class Program19 extends Applet {
        // ορισμός ενός String
        String topLeft = new String("Top Left");
        // ορισμός ενός StringBuffer
        StringBuffer topRight = new StringBuffer("Top Right");

        public void paint(Graphics g) {
                // εμφάνιση του κειμένου του String πάνω αριστερά
                g.drawString(topLeft, 0, 10);
                // εμφάνιση του κειμένου του StringBuffer πάνω δεξιά
                // αφού πρώτα γίνει String με τη μέθοδο toString()
                g.drawString(topRight.toString(), 440, 10);
                // εμφάνιση ενός κειμένου κάτω αριστερά
                g.drawString("Bottom Left", 0, 200);
                // εμφάνιση ενός κειμένου κάτω δεξιά
                g.drawString("Bottom Right", 420, 200);
                // εμφάνιση ενός κειμένου στη μέση
                g.drawString("Middle", 200, 100);
        }
}
```

Για να εμφανίσουμε ένα κείμενο σ' ένα applet χρησιμοποιούμε τη μέθοδο *drawString()*, η οποία δέχεται τρεις παραμέτρους : το κείμενο και τις συντεταγμένες x και y, που καθορίζουν τη θέση του κειμένου. Οι συντεταγμένες x και y αρχίζουν από την πάνω αριστερή γωνία του παραθύρου και αναφέρονται στην κάτω αριστερή γωνία του κειμένου.

Για παράδειγμα, η εντολή *g.drawString("Φλώρινα", 100, 50)* εμφανίζει το κείμενο αρχίζοντας από το σημείο που απέχει 100 κουκκίδες από την αριστερή πλευρά του παραθύρου και 50 κουκκίδες από την πάνω. Το πλάτος και

το μήκος της περιοχής του applet έχουν καθορισθεί στο αρχείο HTML και στη γραμμή που καλεί το applet.

Το κείμενο μπορεί να είναι κάποιο σταθερό κείμενο σε διπλά εισαγωγικά, μια μεταβλητή τύπου String ή οποιοδήποτε άλλο αντικείμενο ή τιμή που μπορεί να μετατραπεί σε κείμενο. Για να εμφανίσουμε ένα αντικείμενο StringBuffer, θα πρέπει πρώτα να το μετατρέψουμε σε String με τη μέθοδο toString(), ως εξής :

```
g.drawString(sBuffer.toString(), 200, 100);
```

Πρόγραμμα 20 – Γραμματοσειρές σε Applet

Στη Java μπορούμε να επιλέξουμε τη γραμματοσειρά που θα χρησιμοποιήσουμε καθώς και το μέγεθος και τη μορφοποίησή της. Μπορούμε να δημιουργήσουμε ένα δικό μας αντικείμενο Font, που έχει τρεις ιδιότητες, το όνομα, το στυλ και το μέγεθος της γραμματοσειράς.

Το στυλ γραμματοσειράς που ορίζουμε παραμένει ενεργό μέχρι να ορίσουμε ένα άλλο. Το παρακάτω πρόγραμμα δημιουργεί αντικείμενα γραμματοσειρών και μετά τα χρησιμοποιεί διαδοχικά για να αλλάξει την εμφάνιση του κειμένου.

```
import java.awt.*;
import java.applet.*;

public class Program20 extends Applet {
        // εδώ ορίζουμε τα τρία αντικείμενα τύπου γραμματοσειράς
        // ο κωδικός 3 σημαίνει έντονη και πλάγια γραφή
        Font header = new Font("TimesRoman", 3, 24);
        Font subhead = new Font("Helvetica", Font.BOLD, 18);
        // ο κωδικός 0 σημαίνει απλή γραφή
        Font body = new Font("Courier", 0, 14);

        public void paint(Graphics g) {
                // ενεργοποίηση της γραμματοσειράς
                // με τη μέθοδο setFont()
                g.setFont(header);
                g.drawString("TimesRoman, έντονο και πλάγιο, 24", 0, 30);
                g.setFont(subhead);
                g.drawString("Helvetica, έντονο, 18", 0, 60);
                g.setFont(body);
                g.drawString("Courier, απλό, 14", 0, 90);
        }
}
```

Πρόγραμμα 21 – Χρώματα σε Applet

Στη Java μπορούμε να ορίσουμε δικά μας αντικείμενα χρώματος μέσω της τάξης Color και να τα χρησιμοποιήσουμε (εφαρμόσουμε) με τη μέθοδο *setColor()* για το χρώμα του κειμένου και των σχημάτων.

Στο παρακάτω πρόγραμμα ορίζουμε τέσσερα αντικείμενα για τα τρία βασικά χρώματα (red, green, blue) και για μια απόχρωση του καφέ και μετά τα εφαρμόζουμε για την εμφάνιση διαφόρων κειμένων.

```
import java.awt.*;
import java.applet.*;

public class Program21 extends Applet {
        // εδώ ορίζουμε τα 4 αντικείμενα τύπου χρώματος
        Color red = new Color(255, 0, 0);
        Color green = new Color(0, 255, 0);
        Color blue = new Color(0, 0, 255);
        Color brown = new Color(80, 64, 0);
        Font sansbold = new Font("Arial", Font.BOLD, 24);

        public void paint(Graphics g) {
                // ενεργοποίηση της γραμματοσειράς
                g.setFont(sansbold);
                // ενεργοποίηση του χρώματος
                g.setColor(red);
                // εμφάνιση του κειμένου με τη συγκεκριμένη
                // γραμματοσειρά και χρώμα
                g.drawString("Κόκκινο", 0, 30);
                g.setColor(green);
                g.drawString("Πράσινο", 0, 60);
                g.setColor(blue);
                g.drawString("Μπλε", 0, 90);
                g.setColor(brown);
                g.drawString("Καφέ", 0, 120);
        }
}
```

Πρόγραμμα 22 – Χρήση του Ποντικιού σε Applet

Για να ελέγξουμε το ποντίκι σ' ένα applet χρησιμοποιούμε τις μεθόδους *mouseDown()*, *mouseUp()* και *mouseDrag()*, που ενεργοποιούνται από την αντίστοιχη ενέργεια του ποντικιού, δηλ. πάτημα του ποντικιού, απελευθέρωση του ποντικιού και σύρσιμο του ποντικιού, λαμβάνοντας υπόψη τις τρέχουσες συντεταγμένες του δείκτη του ποντικιού.

Το παρακάτω πρόγραμμα ανταποκρίνεται σ' ένα απλό κλικ του ποντικιού, εμφανίζοντας τις τρέχουσες συντεταγμένες του δείκτη και όταν το ποντίκι μετακινείται και κρατάμε πατημένο το αριστερό του πλήκτρο, σχεδιάζει μια γραμμή και εμφανίζει τις συντεταγμένες τέλους της γραμμής.

Η λογική (boolean) μεταβλητή drawing χρησιμοποιείται για να ελέγχουμε αν έχει γίνει μετακίνηση (drag) του ποντικιού και μόνο τότε να σχεδιάζεται η γραμμή.

```
import java.awt.*;
import java.applet.Applet;

public class Program22 extends Applet {
        int mx, my;             // η αρχική θέση του δείκτη του ποντικιού
        int mx1, my1;           // η νέα θέση του δείκτη του ποντικιού
        boolean drawing = false;

        public boolean mouseDown(Event e, int x, int y) {
                // αποθήκευση των συντεταγμένων mx και my
                // όταν γίνεται κλικ με το ποντίκι
                mx = x;
                my = y;
                // επανασχεδίαση της οθόνης
                repaint();
                return(true);
        }

        public boolean mouseDrag(Event e, int x, int y) {
                // αποθήκευση των συντεταγμένων mx1 και my1
                // όταν γίνεται σύρσιμο με το ποντίκι
                mx1 = x;
                my1 = y;
                // η λογική μεταβλητή drawing γίνεται ίση με true
                drawing = true;
                // επανασχεδίαση της οθόνης
                repaint();
                return(true);
        }

        public void paint(Graphics g) {
                // εμφάνιση των συντεταγμένων της θέσης του ποντικιού
                g.drawString(mx+", "+my, mx, my);
                // η εντολή if εκτελείται μόνο αν έγινε σύρσιμο
                // του ποντικιού
                if (drawing == true) {
                        // σχεδίαση μιας γραμμής
                        g.drawLine(mx, my, mx1, my1);
                        // εμφάνιση των συντεταγμένων της θέσης
                        // του ποντικιού
                        g.drawString(mx1+", "+my1, mx1, my1);
                } // τέλος του if
                drawing = false;
        }
}
```

Πρόγραμμα 23 – Οι Παράμετροι των Applets

Στη Java μπορούμε να μεταβιβάσουμε τιμές από μια ιστοσελίδα HTML σ' ένα applet και έτσι να μπορούμε να λαμβάνουμε διαφορετικά αποτελέσματα από το ίδιο applet χωρίς να χρειάζεται να κάνουμε αλλαγές σ' αυτό και να το μεταγλωττίζουμε εκ νέου.

Στο παράδειγμα που θα δούμε παρακάτω, το applet μετακινεί ένα μήνυμα κατά μήκος του παραθύρου, όπου το περιεχόμενο του μηνύματος και η ταχύτητα μετακίνησης καθορίζονται με τιμές που δίνονται μέσα από την ιστοσελίδα HTML.

Οι τιμές μεταβιβάζονται μέσω παραμέτρων που γράφονται στην ιστοσελίδα HTML με τη χρήση της ετικέτας <*param*> και λαμβάνονται από το applet μέσω της μεθόδου *getParameter()*.

Οι ετικέτες για τις παραμέτρους γράφονται μέσα στο τμήμα του κώδικα που ορίζουν οι ετικέτες <applet> και έχουν την εξής μορφή :

<param name=όνομα_παραμέτρου value=τιμή_παραμέτρου>

Το *όνομα_παραμέτρου* καθορίζει την ταυτότητά της και μπορούμε να έχουμε όσες παραμέτρους θέλουμε σ' ένα applet. Η *τιμή_παραμέτρου* μπορεί να είναι κείμενο ή αριθμός. Αν είναι κείμενο με περισσότερες από μία λέξεις, τότε πρέπει να τοποθετηθεί μέσα σε διπλά εισαγωγικά.

Ο παρακάτω HTML κώδικας δημιουργεί τις παραμέτρους message και limit για χρήση από ένα applet.

```
<html>
<head>
        <title>Σελίδα Εμφάνισης Applet</title>
</head>
<body>
        <p>Έξοδος Applet :</p>
        <p><applet code="Program23.class" width=500 height=200>
                <param name=message value="Φλώρινα-2008">
                <param name=limit value="1000">
        </applet></p>
</body>
</html>
```

Το πρόγραμμα applet είναι το εξής :

```
import java.awt.*;
import java.applet.*;
```

```
public class Program23 extends Applet {
        // ορίζουμε μια μεγάλη και έντονη γραμματοσειρά
        Font header = new Font("SansSerif", Font.BOLD, 24);
        String text, temp;
        int delayLimit;         // καθυστέρηση

        public void init() {
                // λήψη της τιμής της παραμέτρου για το κείμενο
                text = getParameter("message");
                if (text == null)
                        text = "Κανένα μήνυμα";     // προκαθορισμένη τιμή
                // λήψη της τιμής της παραμέτρου για την καθυστέρηση
                temp = getParameter("limit");
                if (temp == null)
                        delayLimit = 10000; // προκαθορισμένη τιμή
                else
                        delayLimit = Integer.parseInt(temp);
                        // μετατρέπει το String σε Int
        }

        public void paint(Graphics g) {
                // ενεργοποίηση της γραμματοσειράς
                g.setFont(header);
                for (int x = 200; x >0; x--) {
                        // κενός βρόχος για καθυστέρηση
                        for (int delay = 0; delay < delayLimit; delay++)
                                ;
                        // χρήση ενός γκρίζου χρώματος
                        g.setColor(new Color(160, 160, 160));
                        // σχεδίαση ενός συμπαγούς παραλληλογράμμου
                        g.fillRect(x, 80, 300, 30);
                        // χρήση του μαύρου χρώματος
                        g.setColor(new Color(0, 0, 0));
                        // εμφάνιση του κειμένου
                        g.drawString(text, x, 100);
                } // τέλος του for
        }
}
```

Η μέθοδος *getParameter()* παίρνει τις τιμές των παραμέτρων από τον HTML κώδικα και χρησιμοποιείται ως εξής :

```
text = getParameter("message");
```

Το όνομα της παραμέτρου θα πρέπει να γράφεται πάντα μέσα σε διπλά εισαγωγικά και η μέθοδος επιστρέφει πάντα μια μεταβλητή τύπου String.

Αν, όμως, πρόκειται για αριθμό, για να τον μετατρέψουμε σε ακέραιο, θα πρέπει να κάνουμε τα εξής :

```
temp = getParameter("limit");
delayLimit = Integer.parseInt(temp);
```

Με την εντολή *g.setColor(new Color(160, 160, 160))* μπορούμε να δημιουργήσουμε ένα προσωρινό αντικείμενο χρώματος και την χρησιμοποιούμε όταν θέλουμε να ορίσουμε ένα χρώμα για μία μόνο φορά χωρίς να χρειαστεί να ορίσουμε κάποια μεταβλητή χρώματος.

Πρόγραμμα 24 – Τα Πλήκτρα Εντολής σε Applet - 1ο

Στο παρακάτω παράδειγμα χρήσης ενός πλήκτρου εντολής, το πάτημα του πλήκτρου εμφανίζει ένα μήνυμα στην οθόνη και επειδή είναι το μοναδικό γεγονός που μπορεί να συμβεί, δεν ελέγχουμε για κάποιο άλλο γεγονός. Η εντολή *repaint()* προκαλεί την αναγκαστική εκτέλεση της μεθόδου paint().

```
import java.awt.*;
import java.applet.*;

public class Program24 extends Applet {
        // ορισμός ενός αντικειμένου Button
        Button btnClick;
        String message = "";

        public void init() {
                // προσθήκη του αντικειμένου Button στο
                // παράθυρο του applet
                Button btnClick = new Button("Κάντε κλικ εδώ");
                add(btnClick);
        }

        public boolean action(Event evt, Object arg) {
                message = "Florina per sempre!";
                // επανασχεδίαση της οθόνης
                repaint();
                return(true);
        }

        public void paint(Graphics g) {
                // ορισμός ενός αντικειμένου γραμματοσειράς
                g.setFont(new Font("SansSerif", Font.ITALIC, 30));
                // ορισμός ενός αντικειμένου χρώματος
                g.setColor(new Color(160, 0, 0));
                // εμφάνιση του κειμένου
                g.drawString(message, 10, 70);
        }
}
```

Πρόγραμμα 25 – Τα Πλήκτρα Εντολής σε Applet – 2ο

Αν, όμως, έχουμε στην οθόνη μας περισσότερα από ένα πλήκτρα εντολής ή άλλα στοιχεία οθόνης, τότε θα πρέπει να ελέγξουμε ποιο γεγονός έλαβε χώρα και αυτό μπορεί να γίνει με την παράμετρο Event της μεθόδου action(). Συγκρίνουμε το στοιχείο evt.target με το όνομα του πλήκτρου ή άλλου στοιχείου της οθόνης.

Στο επόμενο πρόγραμμα έχουμε δύο πλήκτρα εντολής, τα butRed και butGreen, και ένα έγχρωμο παραλληλόγραμμο. Μόνο ένα πλήκτρο εντολής μπορεί να είναι ενεργοποιημένο κάθε φορά και το κλικ σ' ένα απ' αυτά αλλάζει το χρώμα του παραλληλογράμμου και ενεργοποιεί το άλλο πλήκτρο. Για να ενεργοποιήσουμε ή να απενεργοποιήσουμε ένα πλήκτρο εντολής, δίνουμε την τιμή true ή false αντίστοιχα στη μέθοδο *setEnabled()*.

```
import java.awt.*;
import java.applet.*;

public class Program25 extends Applet {
        Button butRed, butGreen;
        Color red = Color.red;
        Color green = Color.green;
        Color current = red;

        public void init() {
                butRed = new Button("ΚΟΚΚΙΝΟ");
                // απενεργοποίηση του πλήκτρου εντολής αρχικά
                butRed.setEnabled(false);
                butGreen = new Button("ΠΡΑΣΙΝΟ");
                add(butRed);
                add(butGreen);
                current = red;          // χρώμα σχεδίασης κόκκινο
        }

        public boolean action(Event evt, Object arg) {
                // αν πατηθεί το κόκκινο πλήκτρο
                if (evt.target == butRed) {
                        butRed.setEnabled(false);
                        butGreen.setEnabled(true);
                        current = red;          // χρώμα σχεδίασης κόκκινο
                } // τέλος του if

                // αν πατηθεί το πράσινο πλήκτρο
                if (evt.target == butGreen) {
                        butRed.setEnabled(true);
                        butGreen.setEnabled(false);
                        current = green;        // χρώμα σχεδίασης πράσινο
                } // τέλος του if
```

```
                repaint();              // ενημέρωση της οθόνης
                return(true);
        }

        public void paint(Graphics g) {
                // ορισμός του χρώματος σχεδίασης
                g.setColor(current);
                // σχεδίαση του παραλληλογράμμου
                g.fillRect(50, 40, 200, 100);
        }
}
```

Πρόγραμμα 26 – Πλαίσια Ελέγχου & Πλήκτρα Επιλογής

Τα πλαίσια ελέγχου (check boxes) και τα πλήκτρα επιλογής (radio buttons) αντιμετωπίζονται με τον ίδιο τρόπο στη Java. Ως γνωστόν, τα πλαίσια ελέγχου μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν ανεξάρτητα από τα άλλα στοιχεία της οθόνης, ενώ τα πλήκτρα επιλογής εμφανίζονται με τη μορφή ομάδων και μόνο ένα πλήκτρο επιλογής σε κάθε ομάδα μπορεί να είναι ενεργοποιημένο.

Τα πλαίσια ελέγχου πρέπει να δηλώνονται σαν αντικείμενα στη μέθοδο init() ή start() και πρέπει να ορισθεί και η αρχική τους κατάσταση. Η παρακάτω εντολή δημιουργεί ένα πλαίσιο ελέγχου για την επιλογή της έντονης γραφής :

```
Checkbox fontBold = new Checkbox("Έντονα", false);
```

Η πρώτη παράμετρος ορίζει την ετικέτα του πλαισίου ελέγχου και η δεύτερη την αρχική του κατάσταση, δηλ. true ή false, με εξ ορισμού επιλεγμένη την false.

Για να προστεθεί το πλαίσιο ελέγχου στο παράθυρο του applet, χρησιμοποιούμε τη μέθοδο add() :

```
add(fontBold);
```

Για τα πλήκτρα επιλογής, που ονομάζονται και αυτά Checkbox στην Java, πρέπει να δημιουργήσουμε πρώτα την ομάδα που θα τα περιέχει, ως εξής :

```
CheckboxGroup fontName = new CheckboxGroup();
Checkbox sans = new Checkbox("Sans Serif", fontName, true);
```

Εδώ έχουμε τρεις παραμέτρους, όπου η δεύτερη είναι το όνομα της ομάδας όπου ανήκει το πλήκτρο επιλογής. Δεν πρέπει να ξεχνάμε ότι μόνο ένα από τα πλήκτρα επιλογής που ανήκουν στην ίδια ομάδα, μπορεί να είναι σε κατάσταση true (ενεργοποιημένο).

Τον έλεγχο της κατάστασης των πλαισίων ελέγχου και των πλήκτρων επιλογής τον κάνουμε από τη μέθοδο γενικής χρήσης action() και ο καλύτερος τρόπος είναι να γράψουμε εντολές if για όλα τα υπάρχοντα αντικείμενα της κατηγορίας αυτής.

Με τη μέθοδο *getState()* μπορούμε να ελέγξουμε αν ένα πλαίσιο ελέγχου είναι ενεργοποιημένο ή όχι, ως εξής :

```
if (fontBold.getState() == true)
```

Για τα πλήκτρα επιλογής, μπορούμε να χρησιμοποιήσουμε και τη μέθοδο *getSelectedCheckbox()*, η οποία αναφέρεται σε μια ομάδα πλήκτρων επιλογής και μας επιστρέφει το όνομα του επιλεγμένου πλήκτρου επιλογής, ως εξής :

```
if (fontName.getSelectedCheckbox() == sans)
        ...
else if (fontName.getSelectedCheckbox() == serif)
        ...
```

Ακολουθεί ένα πρόγραμμα αφιερωμένο στα πλαίσια ελέγχου και στα πλήκτρα επιλογής :

```
import java.awt.*;
import java.applet.*;

public class Program26 extends Applet {
        // ορισμός δύο πλαισίων ελέγχου για έντονη και πλάγια γραφή
        Checkbox fontBold = new Checkbox("Bold", false);
        Checkbox fontItalic = new Checkbox("Italic", false);
        int bold = 0;
        int italic = 0;

        // ορισμός μιας ομάδας πλήκτρων επιλογής
        CheckboxGroup fontName = new CheckboxGroup();
        // ορισμός τριών πλήκτρων επιλογής που ανήκουν στην ίδια
        // ομάδα fontName
        Checkbox sans = new Checkbox("Sans Serif", fontName, true);
        Checkbox serif = new Checkbox("Serif", fontName, false);
        Checkbox mono = new Checkbox("Monospaced", fontName, false);
        // αρχικά θεωρείται επιλεγμένο το πλήκτρο επιλογής
        // Sans Serif
        String fName = new String("Sans Serif");

        public void init() {
                // προσθέτουμε όλα τα πλαίσια ελέγχου και τα πλήκτρα
                // επιλογής, αλλά όχι και την ομάδα πλήκτρων επιλογής
                add(fontBold);
                add(fontItalic);
```

```
            add(sans);
            add(serif);
            add(mono);
        }

        public boolean action(Event evt, Object arg) {
            // έλεγχος αν είναι ενεργό το πλαίσιο ελέγχου bold
            if (fontBold.getState() == true)
                bold = 1;
            else
                bold = 0;

            // έλεγχος αν είναι ενεργό το πλαίσιο ελέγχου italic
            if (fontItalic.getState() == true)
                italic = 2;
            else
                italic = 0;

            // έλεγχος ποιο πλήκτρο επιλογής είναι επιλεγμένο
            if (fontName.getSelectedCheckbox() == sans)
                fName = "Sans Serif";
            else if (fontName.getSelectedCheckbox() == serif)
                fName = "Serif";
            else
                fName = "Monospaced";

            repaint();              // ενημέρωση της οθόνης
            return(true);
        }

        public void paint(Graphics g) {
            // ορισμός μιας γραμματοσειράς μεγέθους 18 στιγμών,
            // αλλά με τύπο γραμματοσειράς όπως έχει επιλεγεί
            // από τον χρήστη και μορφή 0=απλή, 1=έντονη,
            // 2=πλάγια και 3=έντονη και πλάγια
            g.setFont(new Font(fName, bold+italic, 18));
            // εμφάνιση κειμένου στην οθόνη
            g.drawString("Γραμματοσειρά : "+fName, 30, 50);
        }
    }
```

Το παραπάνω πρόγραμμα χρησιμοποιεί δύο πλαίσια ελέγχου με τις ετικέτες Bold και Italic, για να επιλέξουμε έντονη ή πλάγια γραφή ή και τα δύο και τρία πλήκτρα επιλογής με τις ετικέτες Sans Serif, Serif και Monospaced, για να επιλέξουμε μία μόνο γραμματοσειρά.

Πρόγραμμα 27 – Πλαίσια και Περιοχές Κειμένου – 1ο

Για να δημιουργήσουμε ένα *πλαίσιο κειμένου (TextField)*, μπορούμε να χρησιμοποιήσουμε έναν από τους εξής τρόπους :

- *new TextField()*, που πρέπει να ακολουθείται από μια μέθοδο *setText()* ή *setColumns()* για να ορισθεί το μέγεθος του πλαισίου.
- *new Textfield(int)*, όπου η ακέραια τιμή ορίζει το μέγεθος του πλαισίου σε στήλες.
- *new Textfield(string)*, όπου το string είναι η προκαθορισμένη τιμή του πλαισίου.
- *new Textfield(string, int)*, όπου το string είναι η προκαθορισμένη τιμή του πλαισίου και η ακέραια τιμή το μέγεθός του.

Για να δημιουργήσουμε μια *περιοχή κειμένου (TextArea)*, χρησιμοποιούμε τους ίδιους ορισμούς αλλά και δύο επιπλέον ακέραιες τιμές που ορίζουν το πλάτος και το ύψος της περιοχής. Υπάρχει και ένας πέμπτος τρόπος :

new TextArea(String, int, int, int)

όπου η τελευταία ακέραια τιμή ελέγχει τις γραμμές κύλισης, δηλ. με τιμή 1 απενεργοποιεί την οριζόντια γραμμή κύλισης, με τιμή 2 απενεργοποιεί την κατακόρυφη γραμμή κύλισης και με τιμή 3 απενεργοποιεί και τις δύο.

Στο επόμενο πρόγραμμα δημιουργούμε ένα πλαίσιο κειμένου και μια περιοχή κειμένου. Το πλαίσιο κειμένου email συνοδεύεται από μια ετικέτα (Label) για να εξηγεί στον χρήστη τι πρέπει να γράψει.

Στην περιοχή κειμένου details το κείμενο προτροπής έχει γραφεί μέσα στην περιοχή κειμένου και στη συνέχεια επιλέγεται ολόκληρο με τη μέθοδο *selectAll()* έτσι ώστε να μπορεί να διαγραφεί εύκολα και να αντικατασταθεί από το κείμενο που θα γράψουμε.

```
import java.awt.*;
import java.applet.*;

public class Program27 extends Applet {
        TextField email;        // δήλωση του πλαισίου κειμένου
        TextArea details;       // δήλωση της περιοχής κειμένου

        public void init() {
                // προσθήκη της ετικέτας
                add(new Label("Γράψτε το e-mail : "));
                // προσθήκη του πλαισίου κειμένου
                email = new TextField(40);
                add(email);
                // προσθήκη της περιοχής κειμένου
                details = new TextArea("Όνομα και διεύθυνση", 6, 40);
                add(details);
```

```
                // επιλογή όλου του κειμένου της περιοχής κειμένου
                details.selectAll();
                validate();             // ενημέρωση της οθόνης
        }
}
```

Πρόγραμμα 28 – Πλαίσια και Περιοχές Κειμένου – 2ο

Στο επόμενο πρόγραμμα δημιουργούμε δύο περιοχές κειμένου και ένα πλήκτρο εντολής. Όταν κάνουμε κλικ στο πλήκτρο εντολής, ό,τι κείμενο έχουμε επιλέξει με το ποντίκι στην αριστερή περιοχή κειμένου αντιγράφεται στη δεξιά περιοχή κειμένου.

Οι μέθοδοι που χρησιμοποιούμε είναι η *getSelectedText()*, που επιστρέφει το επιλεγμένο κείμενο, η *getText()*, που επιστρέφει όλο το κείμενο, η *setText()*, που ορίζει το κείμενο που αντικαθιστά οποιοδήποτε υπάρχον κείμενο και η *append()*, που προσθέτει ένα τμήμα κειμένου.

```
import java.awt.*;
import java.applet.*;

public class Program28 extends Applet {
        TextArea source;            // δήλωση των δύο περιοχών κειμένου
        TextArea destination;
        Button copy;                // δήλωση του πλήκτρου εντολής

        public void init() {
                // προσθήκη της πρώτης περιοχής κειμένου
                source = new TextArea(10, 30);
                add(source);
                // προσθήκη του πλήκτρου εντολής
                copy = new Button("Αντιγραφή επιλεγμένου κειμένου");
                add(copy);
                // προσθήκη της δεύτερης περιοχής κειμένου
                destination = new TextArea(10, 30);
                add(destination);
                validate();             // ενημέρωση της οθόνης
        }

        public boolean action(Event evt, Object arg) {
                String temp;

                // έλεγχος αν έγινε κλικ στο πλήκτρο εντολής
                if (evt.target == copy) {
                        // καταχώρηση στη μεταβλητή temp
                        // του επιλεγμένου κειμένου
                        // από την πρώτη περιοχή κειμένου
                        temp = source.getSelectedText();
                        // καταχώρηση ή προσθήκη του επιλεγμένου
```

```
                    // κειμένου στη δεύτερη περιοχή κειμένου
                    if (destination.getText() == "")
                        destination.setText(temp);
                    else
                        destination.append(temp);
                } // τέλος του πρώτου if
                return true;
        }
}
```

Πρόγραμμα 29 – Πλαίσια και Περιοχές Κειμένου – 3ο

Επειδή τα στοιχεία κειμένου της Java λαμβάνουν βασικά μόνο κείμενο (string), όταν θέλουμε να δουλέψουμε με αριθμούς πρέπει να κάνουμε κάποιες μετατροπές. Στην τάξη Integer υπάρχει η μέθοδος *parseInt()*, που μετατρέπει ένα string σε ακέραιο αριθμό.

Για να πάρουμε λοιπόν την ακέραια τιμή ενός πλαισίου κειμένου, με το όνομα π.χ. entry, πρέπει πρώτα να πάρουμε το κείμενο και να το τοποθετήσουμε σε μια μεταβλητή temp τύπου string και μετά με τη μέθοδο parseInt() να μετατρέψουμε το temp σε ακέραιο αριθμό, ως εξής :

```
temp = entry.getText();
number = Integer.parseInt(temp);
```

Το επόμενο πρόγραμμα παράγει έναν τυχαίο ακέραιο αριθμό από 0 έως 99 και προκαλεί τον χρήστη να τον βρει γράφοντας τιμές σ' ένα πεδίο κειμένου και κάνοντας κλικ σ' ένα πλήκτρο εντολής. Εμφανίζονται τα μηνύματα «πολύ μεγάλος» ή «πολύ μικρός» σε μια ετικέτα, μέχρι να βρεθεί ο αριθμός.

```
import java.awt.*;
import java.applet.*;

public class Program29 extends Applet {
        Button btnClick;        // δήλωση πλήκτρου εντολής
        TextField entry;        // δήλωση πλαισίου κειμένου
        Label prompt;           // δήλωση ετικέτας
        String temp;            // δήλωση μεταβλητής String
        // παραγωγή τυχαίου ακεραίου αριθμού στο διάστημα 0-99
        int x = (int)(java.lang.Math.random() * 100);
        int number;             // δήλωση ακέραιας μεταβλητής

        public void init() {
                // προσθήκη της ετικέτας
                prompt = new Label("Μάντεψε τον αριθμό : ");
                add(prompt);
                // προσθήκη του πλαισίου κειμένου
                entry = new TextField(10);
```

```
            add(entry);
            // προσθήκη του πλήκτρου εντολής
            btnClick = new Button("΄Ελεγξε την επιλογή σου");
            add(btnClick);
        }

        public boolean action(Event evt, Object arg) {
            // καταχώρηση στη μεταβλητή temp του κειμένου
            // που υπάρχει στο πλαίσιο κειμένου
            temp = entry.getText();
            // μετατροπή του κειμένου σε ακέραιο αριθμό
            number = Integer.parseInt(temp);
            // έλεγχος του αριθμού κάνοντας σύγκριση με τον
            // τυχαίο αριθμό x
            if (x > number)
                prompt.setText("Πολύ μικρός. Δοκίμασε πάλι");
            else if (x < number)
                prompt.setText("Πολύ μεγάλος. Δοκίμασε πάλι");
            else
                prompt.setText("Μπράβο. Τον βρήκες!");
            return true;
        }
}
```

Πρόγραμμα 30 – Σύνθετα Πλαίσια και Λίστες

Το *σύνθετο πλαίσιο (choice)* είναι μια πτυσσόμενη λίστα από την οποία μπορούμε να επιλέξουμε μόνο ένα αντικείμενο, ενώ μια *λίστα (list)* είναι ένα πλαίσιο με κατακόρυφη γραμμή κύλισης και μπορούμε να επιλέξουμε περισσότερα από ένα αντικείμενα ταυτόχρονα.

Αν και αυτά τα δύο αντικείμενα ορίζονται με παρόμοιες ρουτίνες, χρειάζεται διαφορετική προσέγγιση για τον έλεγχο και την καταγραφή των επιλογών που γίνονται σ' αυτά.

΄Ενα σύνθετο πλαίσιο ή μια λίστα μπορεί να οριστεί ως εξής :

```
Choice c = new Choice();
c.addItem("Επιλογή1");
c.addItem("Επιλογή2");
add(c);
```

Οι επιλογές εμφανίζονται στη λίστα με τη σειρά που έχουν προστεθεί μέσω της μεθόδου *addItem()* και παίρνουν έναν αύξοντα αριθμό ή αριθμό ευρετηρίου, ξεκινώντας από το 0. Για να θέσουμε ως προκαθορισμένη μια επιλογή, χρησιμοποιούμε τη μέθοδο *select()* και τον αριθμό ευρετηρίου της επιλογής, ως εξής :

```
c.select(1);            // επιλέγεται η 2η καταχώρηση
```

Για να βρούμε ποια καταχώρηση είναι επιλεγμένη, χρησιμοποιούμε τη μέθοδο *getSelectedIndex()*, ως εξής :

```
chosen = c.getSelectedIndex();
```

ή και ως εξής :

```
switch (c.getSelectedIndex()) {
        case 0 : ... ; break;
        case 1 : ... ; break;
        ...
}
```

Η μέθοδος getSelectedIndex() χρησιμοποιείται λιγότερο στις λίστες, στις οποίες μπορούν να επιλεγούν περισσότερες από μία καταχωρήσεις κάθε φορά. Μπορούμε βέβαια να χρησιμοποιήσουμε τη μέθοδο *setMultipleMode (false)* για να γίνεται μία μόνο επιλογή στη λίστα.

Για να βρούμε αν είναι επιλεγμένη κάποια καταχώρηση, χρησιμοποιούμε τη μέθοδο *isIndexSelected()*, στην οποία δίνουμε τον αριθμό ευρετηρίου του αντικειμένου και μας επιστρέφει την τιμή true αν η καταχώρηση είναι επιλεγμένη.

Το επόμενο πρόγραμμα μάς επιτρέπει να επιλέξουμε τα χαρακτηριστικά του υπολογιστή μας και χρησιμοποιεί ένα σύνθετο πλαίσιο με το όνομα micro, που περιέχει τους τύπους των μικροεπεξεργαστών (P133, P150, P166 και P200), και μια λίστα με το όνομα parts, που περιέχει τα διάφορα περιφερειακά και τις μονάδες μνήμης.

Υπάρχει ακόμα ένας πίνακας τεσσάρων θέσεων με τους τύπους των μικροεπεξεργαστών και ένας άλλος πίνακας τεσσάρων θέσεων με τις αντίστοιχες τιμές τους. Αφού κάνουμε τις επιλογές μας και πατήσουμε το πλήκτρο εντολής με τον τίτλο OK, το πρόγραμμα εμφανίζει σε μια διπλανή περιοχή κειμένου τις επιλογές που έχουμε κάνει.

```
import java.awt.*;
import java.applet.*;

public class Program30 extends Applet {
        Choice micro;          // δήλωση σύνθετου πλαισίου
        List parts;            // δήλωση λίστας
        Button done;           // δήλωση πλήκτρου εντολής
        TextArea spec;         // δήλωση περιοχής κειμένου

        // πίνακας των τύπων των επεξεργαστών
        int[ ] procType = new int[4];
        // πίνακας των τιμών των επεξεργαστών
        float[ ] procCost = new float[4];
        int p = 0;      // δήλωση ακέραιας μεταβλητής
```

```
// πίνακας των περιφερειακών
String[ ] peripheral = new String[8];

public void init() {
        // προσθήκη του σύνθετου πλαισίου με τις τιμές του
        micro = new Choice();
        micro.addItem("P133");
        micro.addItem("P150");
        micro.addItem("P166");
        micro.addItem("P200");
        // επιλογή της πρώτης καταχώρησης (0)
        micro.select(0);
        add(micro);

        // καταχώρηση τιμών στους δύο πίνακες
        // για τον τύπο και την τιμή των επεξεργαστών
        procType[0] = 133;
        procCost[0] = 85;
        procType[1] = 150;
        procCost[1] = 95;
        procType[2] = 166;
        procCost[2] = 135;
        procType[3] = 200;
        procCost[3] = 165;

        // καταχώρηση τιμών στον πίνακα των περιφερειακών
        peripheral[0] = "CD-ROM";
        peripheral[1] = "Κάρτα ΄Ηχου";
        peripheral[2] = "Μεγάφωνα";
        peripheral[3] = "Tape Drive";
        peripheral[4] = "Zip Drive";
        peripheral[5] = "Modem";
        peripheral[6] = "Extra 16MB RAM";
        peripheral[7] = "1,2 GB Σκληρός Δίσκος";

        // προσθήκη λίστας με τιμές τα περιεχόμενα
        // του πίνακα των περιφερειακών
        parts = new List(6, true);
        for (int loop = 0; loop < 8; loop++)
                parts.addItem(peripheral[loop]);
        add(parts);

        // προσθήκη πλήκτρου εντολής
        done = new Button("OK");
        add(done);
        // προσθήκη περιοχής κειμένου
        spec = new TextArea(10, 30);
        add(spec);
```

```
                validate();
                // ενημέρωση της οθόνης μετά την προσθήκη
                // των αντικειμένων
        }

        public boolean action(Event evt, Object arg) {
                // ποιο μοντέλο έχει επιλεγεί από το σύνθετο πλαίσιο
                p = micro.getSelectedIndex();
                // έλεγχος αν έχει πατηθεί το πλήκτρο εντολής
                if (evt.target == done) {
                        // καταχώρηση στην περιοχή κειμένου
                        spec.setText("Pentium "+procType[p]+
                                " στα "+procCost[p]+'\n');
                        // έλεγχος αν είναι επιλεγμένο κάτι από τη λίστα
                        // με τα περιφερειακά και αν ναι προσθήκη του
                        // στην περιοχή κειμένου
                        for (int loop = 0; loop < 8; loop++)
                                if (parts.isIndexSelected(loop) == true)
                                        spec.append(peripheral[loop]+'\n');
                } // τέλος του if
                return true;
        }
}
```

Πρόγραμμα 31 – Σχεδίαση Γραμμής

Η μέθοδος *drawLine()* σχεδιάζει μια γραμμή ανάμεσα σε δύο σημεία.

Για παράδειγμα, η παρακάτω εντολή :

```
g.drawLine(0, 0, 100, 50)
```

σχεδιάζει μια λεπτή γραμμή από την πάνω αριστερή γωνία του παραθύρου μέχρι το σημείο που απέχει 100 κουκκίδες από την αριστερή πλευρά του παραθύρου και 50 από την πάνω πλευρά.

Αν θέλουμε μια πιο πλατιά γραμμή, θα πρέπει να σχεδιάσουμε και μια άλλη γραμμή ακριβώς δίπλα στην προηγούμενη. Αυτό ακριβώς κάνει το επόμενο πρόγραμμα.

```
import java.awt.*;
import java.applet.*;

public class Program31 extends Applet {
        public void paint(Graphics g) {
                // σχεδίαση μιας γραμμής με μαύρο χρώμα
                g.drawLine(0, 0, 100, 50);
                // ορισμός του κόκκινου ως το χρώμα σχεδίασης
                g.setColor(new Color(255, 0, 0));
```

```
                // σχεδίαση μιας ακόμα γραμμής
                g.drawLine(50, 75, 200, 25);
                // σχεδίαση μιας γραμμής δίπλα στην προηγούμενη
                g.drawLine(50, 76, 200, 26);
                // ορισμός του πράσινου ως το χρώμα σχεδίασης
                g.setColor(new Color(0, 255, 0));

                // σχεδιάζουμε 10 γραμμές τη μια δίπλα στην άλλη
                for (int y1 = 120; y1 < 130; y1++)
                        g.drawLine(100, y1, 200, y1+20);
        }
}
```

Πρόγραμμα 32 – Σχεδίαση Παραλληλογράμμου

Για να σχεδιάσουμε παραλληλόγραμμα, υπάρχει η μέθοδος *drawRect()*, που σχεδιάζει το περίγραμμα ενός παραλληλογράμμου και η *fillRect()*, που δημιουργεί ένα συμπαγές παραλληλόγραμμο. Και στις δύο μεθόδους πρέπει να δοθούν οι συντεταγμένες της πάνω αριστερά γωνίας και οι διαστάσεις του παραλληλογράμμου, ως εξής :

```
g.fillRect(50, 150, 200, 100);
```

Η παραπάνω εντολή σχεδιάζει ένα συμπαγές παραλληλόγραμμο με πλάτος 200, ύψος 100 και σημείο εκκίνησης (πάνω αριστερά) το 50, 150.

Οι γραμμές που σχεδιάζει η μέθοδος drawRect() είναι λεπτές και αν θέλουμε ένα πλατύτερο πλαίσιο, μπορούμε να χρησιμοποιήσουμε έναν βρόχο για να σχεδιάσουμε πολλά πλαίσια ή να τοποθετήσουμε ένα fillRect() μέσα σ' ένα άλλο. Αυτό ακριβώς κάνει το επόμενο πρόγραμμα.

```
import java.awt.*;
import java.applet.*;

public class Program32 extends Applet {
        // δήλωση μιας μεταβλητής για το κόκκινο χρώμα
        Color red = new Color(255, 0, 0);
        // δήλωση μιας μεταβλητής για το κίτρινο χρώμα
        Color yellow = new Color(255, 255, 0);
        // δήλωση μιας μεταβλητής για το μαύρο χρώμα
        Color black = new Color(0, 0, 0);
        // δήλωση μιας μεταβλητής γραμματοσειράς
        Font text = new Font("SansSerif", Font.BOLD, 18);

        public void paint(Graphics g) {
                // ορισμός της γραμματοσειράς κειμένου
                g.setFont(text);
                // ορισμός του κόκκινου χρώματος για τη σχεδίαση
                g.setColor(red);
```

```
                // σχεδιάζουμε 5 παραλληλόγραμμα το ένα μέσα στο άλλο
                for (int loop = 0; loop < 5; loop++)
                    g.drawRect(50+loop, 50+loop, 150-loop*2, 75-loop*2);
                // ορισμός του μαύρου χρώματος για τη σχεδίαση
                g.setColor(black);
                // εμφάνιση ενός κειμένου μέσα στο κόκκινο πλαίσιο
                g.drawString("Πλαίσιο...", 80, 90);
                // ορισμός του κόκκινου χρώματος για τη σχεδίαση
                g.setColor(red);
                // σχεδίαση ενός συμπαγούς παραλληλογράμμου
                g.fillRect(50, 150, 200, 100);
                // ορισμός του κίτρινου χρώματος για τη σχεδίαση
                g.setColor(yellow);
                // σχεδίαση ενός συμπαγούς παραλληλογράμμου
                // μέσα στο προηγούμενο παραλληλόγραμμο
                g.fillRect(60, 160, 180, 80);
                // ορισμός του μαύρου χρώματος για τη σχεδίαση
                g.setColor(black);
                // εμφάνιση ενός κειμένου μέσα στο κίτρινο πλαίσιο
                g.drawString("...πάλι!", 100, 200);
        }
}
```

Πρόγραμμα 33 – Σχεδίαση ΄Ελλειψης

Για να σχεδιάσουμε ελλείψεις και κύκλους, μπορούμε να χρησιμοποιήσουμε τις μεθόδους *drawOval()* και *fillOval()*, όπου ορίζουμε τις συντεταγμένες της πάνω αριστερής γωνίας και τις διαστάσεις του παραλληλογράμμου που τις περικλείει. Στο παρακάτω πρόγραμμα οι ελλείψεις σχεδιάζονται μαζί με τα αντίστοιχα παραλληλόγραμμα που τις περικλείουν.

```
import java.awt.*;
import java.applet.*;

public class Program33 extends Applet {
        // δήλωση μιας μεταβλητής για ένα γκρι χρώμα
        Color grey = new Color(160, 160, 160);
        // δήλωση μιας μεταβλητής για το μαύρο χρώμα
        Color black = new Color(0, 0, 0);

        public void paint(Graphics g) {
                // γκρι έλλειψη εγγεγραμμένη σε
                // ορθογώνιο παραλληλόγραμμο
                g.drawRect(10, 10, 150, 100);
                g.setColor(grey);
                g.fillOval(10, 10, 150, 100);

                // δύο κύκλοι, μαύρος και γκρι, εγγεγραμμένοι σε γκρι
                // ορθογώνιο παραλληλόγραμμο
```

```
                g.fillRect(180, 40, 100, 100);
                g.setColor(black);
                g.fillOval(180, 40, 100, 100);
                g.setColor(grey);
                g.fillOval(200, 60, 60, 60);
        }
}
```

Πρόγραμμα 34 – Σχεδίαση Πολυγώνου

Για να σχεδιάσουμε πολύγωνα, δηλ. κλειστά σχήματα, από τρίγωνα έως σύνθετα σχήματα με οποιονδήποτε αριθμό πλευρών, μπορούμε να χρησιμοποιήσουμε τις μεθόδους *drawPolygon()* και *fillPolygon()*. Τα σχήματα ορίζονται με βάση τις συντεταγμένες των κορυφών τους, οι οποίες γράφονται με τη μορφή ενός ζευγαριού παρόμοιων πινάκων ακεραίων αριθμών.

Οι παραπάνω μέθοδοι παίρνουν ως παραμέτρους τα ονόματα των δύο πινάκων συντεταγμένων και το πλήθος των κορυφών του πολυγώνου. Υπάρχει και η μέθοδος *drawPolyline()*, που χρησιμοποιείται με τον ίδιο τρόπο με την drawPolygon(), αλλά δεν δημιουργεί κλειστό σχήμα.

Το παρακάτω πρόγραμμα δημιουργεί ένα εξάγωνο.

```
import java.awt.*;
import java.applet.*;

public class Program34 extends Applet {
        int[ ] x = new int[6];                  // πίνακας συντεταγμένων x
        int[ ] y = new int[6];                  // πίνακας συντεταγμένων y

        public void init() {
                // απόδοση τιμών στους δύο πίνακες
                x[0] = 100; y[0] = 14;
                x[1] = 200; y[1] = 14;
                x[2] = 250; y[2] = 100;
                x[3] = 200; y[3] = 186;
                x[4] = 100; y[4] = 186;
                x[5] = 50; y[5] = 100;
        }

        public void paint(Graphics g) {
                // σχεδίαση του πολυγώνου (εξαγώνου)
                g.drawPolygon(x, y, 6);
        }
}
```

Πρόγραμμα 35 – Σχεδίαση Τόξου

Οι μέθοδος σχεδίασης τόξου, που είναι οι *drawArc()* και *fillArc()*, είναι παρόμοιες με τις μεθόδους για τον σχεδιασμό ελλείψεων, μόνο που έχουν δύο πρόσθετες παραμέτρους, μια για τον ορισμό της αρχής του τόξου και μια για το μέγεθος της γωνίας του τόξου.

Για παράδειγμα, η παρακάτω εντολή

g.fillArc(10, 10, 100, 100, 0, 90);

σχεδιάζει ένα τόξο κύκλου που ορίζεται από το παραλληλόγραμμο, στην ουσία τετράγωνο, με την πάνω αριστερή γωνία του στη θέση 10, 10 και μέγεθος 100 X 100. Το τόξο αρχίζει από τη θέση 0 μοίρες και φθάνει έως τις 90^o.

Πρέπει να έχουμε υπόψη μας ότι οι γωνίες των τόξων δίνονται σε μοίρες (ακέραιοι αριθμοί), όπου οι 0^o αντιστοιχούν στην ώρα 3 και οι γωνίες αυξάνουν με φορά αντίθετη απ' αυτή των δεικτών του ρολογιού.

Το παρακάτω πρόγραμμα δημιουργεί διάφορα τόξα :

```
import java.awt.*;
import java.applet.*;

public class Program35 extends Applet {
        public void paint(Graphics g) {
                // σχεδίαση κυκλικού τομέα από 0 έως 90 μοίρες
                g.fillArc(10, 10, 100, 100, 0, 90);
                // σχεδίαση κυκλικού τόξου από 90 έως 180 μοίρες
                g.drawArc(10, 10, 100, 100, 90, 180);
                // σχεδίαση κυκλικού τομέα
                g.fillArc(10, 120, 100, 100, 90, -60);
                // σχεδίαση τμήματος έλλειψης
                g.drawArc(120, 120, 100, 50, 270, -210);
        }
}
```

Πρόγραμμα 36 – Σχεδίαση Διαγράμματος Πίτας

Για να σχεδιάσουμε διαγράμματα πίτας ουσιαστικά χρησιμοποιούμε τη μέθοδο σχεδίασης τόξων και μερικές προγραμματιστικές τεχνικές. Στα διαγράμματα πίτας έχουμε ένα σύνολο αριθμών, τους οποίους προσθέτουμε, υπολογίζουμε το ποσοστό % συμμετοχής του κάθε ποσού στο σύνολο και μετά σχεδιάζουμε αντίστοιχα συνεχόμενα τόξα με γέμιση (fillArc) και με τα προκαθορισμένα χρώματα που διαθέτει η Java.

Θα χρειαστούμε έναν πίνακα για τις τιμές του διαγράμματος, έναν για τις μοίρες της κάθε φέτας (τόξου) και έναν για το χρώμα της κάθε φέτας. Τις

τιμές του διαγράμματος τις δίνουμε στη μέθοδο init(), υπολογίζουμε το άθροισμά τους και μετά το μέγεθος κάθε τόξου σε μοίρες.

Στην τάξη java.awt.Color υπάρχουν 13 προκαθορισμένα χρώματα που είναι πολύ χρήσιμα σε καταστάσεις όπως αυτές με τα διαγράμματα πίτας, όπου χρειαζόμαστε μια ποικιλία χρωμάτων χωρίς να μας ενδιαφέρει η ακριβής απόχρωση. Για να τα χρησιμοποιήσουμε, γράφουμε απλά τη λέξη Color και το όνομα του χρώματος, π.χ. :

```
col[0] = Color.red;
```

Τα υπόλοιπα προκαθορισμένα χρώματα είναι τα εξής : black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, white και yellow.

Το πρόγραμμα δημιουργίας ενός διαγράμματος πίτας για 6 κομμάτια είναι το εξής :

```
import java.awt.*;
import java.applet.*;

public class Program36 extends Applet {
        double[ ] value = new double[6];  // τιμές του διαγράμματος
        int[ ] slice = new int[6];        // μοίρες της κάθε φέτας
        Color[ ] col = new Color[6];      // χρώμα της κάθε φέτας

        public void init() {
                double total = 0;         // άθροισμα των τιμών

                // απόδοση των τιμών του διαγράμματος
                value[0] = 1000;
                value[1] = 500;
                value[2] = 750;
                value[3] = 1200;
                value[4] = 300;
                value[5] = 1650;

                // υπολογισμός του αθροίσματος των τιμών
                // του διαγράμματος
                for (int loop = 0; loop < 6; loop++)
                        total += value[loop];
                // υπολογισμός των μοιρών της κάθε φέτας
                // του διαγράμματος
                for (int loop = 0; loop < 6; loop++)
                        slice[loop] = (int) (value[loop] / total * 360);

                // καθορισμός του χρώματος της κάθε φέτας
                col[0] = Color.red;
                col[1] = Color.green;
```

```
            col[2] = Color.blue;
            col[3] = Color.gray;
            col[4] = Color.magenta;
            col[5] = Color.yellow;
        }

        public void paint(Graphics g) {
            int start = 0;          // γωνία εκκίνησης φέτας
            for (int loop = 0; loop < 6; loop++) {
                g.setColor(col[loop]);      // επόμενο χρώμα
                g.fillArc(20, 20, 200, 200, start, slice[loop]);
                start += slice[loop];
                // αλλαγή της γωνίας εκκίνησης της επόμενης φέτας
            } // τέλος του for
        }
    }
```

Πρόγραμμα 37 – Προσθήκη Εικόνας

Στην Java οι εικόνες αποθηκεύονται σε αντικείμενα *Image*, ενώ τα αρχεία φορτώνονται σ' αυτά με τη μέθοδο *getImage()*, ως εξής :

```
Image pic;
pic = getImage(getCodeBase(), "\images\car.gif");
```

Η μέθοδος getImage() παίρνει ως παράμετρο ένα URL, για να μπορέσει να εντοπίσει ένα αρχείο στο Internet ή σ' έναν υπολογιστή. Η μέθοδος *getCodeBase()* προσδιορίζει το URL με βάση τον τρέχοντα φάκελο όπου είναι αποθηκευμένο το πρόγραμμα της Java.

Οι εικόνες μπορούν να τοποθετηθούν κατευθείαν στο applet με τη χρήση της μεθόδου *drawImage()*, που στην απλή της μορφή συντάσσεται ως εξής :

```
g.drawImage(pic, 0, 0 this);
```

Η παραπάνω εντολή σχεδιάζει την εικόνα pic με το πραγματικό της μέγεθος, με την πάνω αριστερή γωνία να είναι στη θέση 0, 0. Η παράμετρος this αναφέρεται στο παράθυρο όπου θα τοποθετηθεί η εικόνα.

Η δεύτερη απλή μορφή της μεθόδου χρησιμοποιεί δύο επιπλέον παραμέτρους για το πλάτος και το ύψος της εικόνας, αν φυσικά αυτά διαφέρουν από τα κανονικά :

```
g.drawImage(pic, 0, 0, 100, 50, this);
```

Το παρακάτω πρόγραμμα κάνει μια εφαρμογή των παραπάνω μεθόδων.

```
import java.awt.*;
import java.applet.*;

public class Program37 extends Applet {
        // δήλωση μιας μεταβλητής εικόνας
        Image pic;

        public void init() {
                // επιλογή της εικόνας από τον τρέχοντα φάκελο
                pic = getImage(getCodeBase(), "car.gif");
        }

        public void paint(Graphics g) {
                // εμφάνιση της εικόνας με το κανονικό της μέγεθος
                g.drawImage(pic, 0, 0, this);
                // εμφάνιση της εικόνας με συγκεκριμένες διαστάσεις
                g.drawImage(pic, 100, 100, 125, 100, this);
        }
}
```

Πρόγραμμα 38 – Προσθήκη Ήχου

Στο επόμενο πρόγραμμα, όταν εκτελεστεί το applet, θα ακουστεί ένα συνεχόμενο μπιπ και θα εμφανισθούν δύο πλήκτρα εντολών. Αν κάνουμε κλικ σ' αυτό που γράφει «Εξυπηρέτηση», θα ακουστεί το αρχείο ding.au μία φορά, ενώ αν κάνουμε κλικ στο «Διακοπή!», θα σταματήσει το μπιπ.

```
import java.awt.*;
import java.applet.*;

public class Program38 extends Applet {
        // δήλώση ενός αντικειμένου τύπου AudioClip
        AudioClip beeper;
        // δήλωση δύο πλήκτρων εντολής
        Button ringer;
        Button quiet;

        public void init() {
                beeper = getAudioClip(getCodeBase(),"beep.au");
                // προσθήκη των δύο πλήκτρων εντολής
                ringer = new Button("Εξυπηρέτηση");
                add(ringer);
                quiet = new Button("Διακοπή!");
                add(quiet);
                // συνεχόμενη εκτέλεση του αρχείου μουσικής
                beeper.loop();
        }
```

```
        public boolean action(Event e, Object obj) {
                // έλεγχος αν πατήθηκε το πλήκτρο εντολής ringer
                if (e.target == ringer)
                        // φόρτωση και εκτέλεση του αρχείου ήχου
                        // μία φορά
                        play(getCodeBase(), "ding.au");
                // έλεγχος αν πατήθηκε το πλήκτρο εντολής quiet
                if (e.target == quiet)
                        // διακοπή της εκτέλεσης του αρχείου μουσικής
                        beeper.stop();
                return true;
        }
}
```

Πρόγραμμα 39 – Κείμενα και Πίνακες

Με τη μέθοδο *toCharArray()* μπορούμε να μετατρέψουμε μια στοιχειοσειρά (string) σ' έναν πίνακα από χαρακτήρες. Τους χαρακτήρες αυτούς μπορούμε μετά να τους επεξεργαστούμε, όπως να ορίσουμε διαφορετικό χρώμα, γραμματοσειρά, μέγεθος ή και θέση στην οθόνη.

Άλλες χρήσιμες μέθοδοι είναι η *length()*, που επιστρέφει το μήκος μιας στοιχειοσειράς, η *getCharWidth()*, που δίνει το πλάτος ενός χαρακτήρα σε κουκκίδες και η *drawChars()*, που εμφανίζει έναν ή περισσότερους χαρακτήρες ενός πίνακα.

Στο παρακάτω πρόγραμμα μετατρέπουμε τη στοιχειοσειρά text στον πίνακα χαρακτήρων letter και μετά τοποθετούμε τα γράμματα σε τυχαίες θέσεις στην οθόνη.

```
import java.awt.*;
import java.applet.*;

public class Program39 extends Applet {
        String text = "Florina";        // δήλωση ενός String
        int slength;                    // μήκος της στοιχειοσειράς
        char[ ] letter;                 // δήλωση ενός πίνακα χαρακτήρων

        public void init() {
                // εύρεση του μήκους της στοιχειοσειράς
                slength = text.length();
                // δημιουργία του πίνακα χαρακτήρων
                letter = new char[slength];
                // μετατροπή του κειμένου σε πίνακα
                letter = text.toCharArray();
        }

        // η παρακάτω μέθοδος εκτελείται όταν πατήσουμε και
        // μετακινήσουμε το ποντίκι, οπότε καλείται η μέθοδος paint()
```

```
        public boolean mouseDrag(Event e, int x, int y) {
                repaint();
                return true;
        }

        public void paint(Graphics g) {
                int x, y;
                x = 0;
                y = 50;
                for (int loop = 0; loop < slength; loop++) {
                        g.drawChars(letter, loop, 1, x, y);
                        x++;
                        // παράγει μια τυχαία ακέραια τιμή στο διάστημα
                        // 50 +/- 10, δηλ. 40 έως 60
                        y = 50 +(int) ((java.lang.Math.random()*20)-10);
                } // τέλος του for
        }
}
```

Η μέθοδος drawChars() παίρνει σαν παραμέτρους το όνομα του πίνακα χαρακτήρων, τη θέση του πρώτου χαρακτήρα, τον αριθμό των χαρακτήρων και τις συντεταγμένες x, y του σημείου τοποθέτησης.

Πρόγραμμα 40 – Δημιουργία Τυχαίων Αριθμών

Το παρακάτω πρόγραμμα Java δημιουργεί 1.000 τυχαίους ακεραίους αριθμούς από το 0 έως το 9 και βρίσκει πόσες φορές εμφανίζεται ο καθένας. Από εδώ μπορούμε να κρίνουμε αν η συνάρτηση java.lang.Math.random() όντως δημιουργεί πραγματικά τυχαίους αριθμούς, αν βέβαια είναι περίπου ίδιο το ποσοστό εμφάνισης του κάθε αριθμού.

```
class Program40 {
        public static void main(String[ ] args) {
                // δήλωση ενός πίνακα ακεραίων
                int[ ] num = new int[10];
                // δήλωση δύο ακεραίων μεταβλητών
                int loop, x;

                // μηδενισμός των τιμών του πίνακα
                for (loop = 0; loop < 10; loop++)
                        num[loop] = 0;

                // δημιουργία ενός τυχαίου ακεραίου αριθμού
                // στο διάστημα 0-9 και αύξηση κατά ένα της
                // αντίστοιχης θέσης του στον πίνακα
                for (loop = 0; loop < 1000; loop++) {
                        x = (int) (java.lang.Math.random()*10);
                        num[x]++;
                } // τέλος του for
```

```
                // εκτύπωση των φορών που εμφανίσθηκε ο κάθε αριθμός
                for (loop = 0; loop < 10; loop++)
                        System.out.println(loop + " φορές " + num[loop]);
        }
}
```

Πρόγραμμα 41 – Εμφάνιση Κειμένου με Applet

Το παρακάτω applet εμφανίζει ένα μήνυμα με μεγάλη γραμματοσειρά και έντονο χρώμα, όπου το χρώμα σβήνει σταδιακά από το σκούρο έως το λευκό.

```
import java.awt.*;
import java.applet.*;

public class Program41 extends Applet {
        int shade;      // δήλωση μιας ακέραιας μεταβλητής
        // δήλωση μιας μεταβλητής γραμματοσειράς
        Font sansbold = new Font("Helvetica", Font.BOLD, 24);

        public void paint(Graphics g) {
                for (shade = 0; shade < 256; shade++) {
                        // χρήση συγκεκριμένης γραμματοσειράς
                        g.setFont(sansbold);
                        // ορισμός ενός γκρι χρώματος σχεδίασης
                        // σταδιακά από τελείως μαύρο έως τελείως λευκό
                        g.setColor(new Color(shade, shade, shade));
                        // εμφάνιση του κειμένου σε σταθερή θέση
                        g.drawString("Σβήσιμο ...", 0, 30);
                        // χρονική καθυστέρηση με βρόχο for
                        for (int delay = 0; delay < 250000; delay++)
                                ;
                } // τέλος του for
        }
}
```

Πρόγραμμα 42 – Αλλαγή Μεγέθους Κειμένου με Applet

Το παρακάτω applet αλλάζει το μέγεθος της γραμματοσειράς ενός μηνύματος, όπου το κλικ στο πάνω μισό τμήμα του παραθύρου μεγαλώνει το μέγεθος και το κλικ στο κάτω μισό το μειώνει. Πρέπει να ορισθεί η παράμετρος height σε 400 στη σελίδα HTML.

```
import java.awt.*;
import java.applet.*;

public class Program42 extends Applet {
        // μεταβλητή για το μέγεθος της γραμματοσειράς
        int size = 24;
```

```
        public boolean mouseDown(Event e, int x, int y) {
                // αν το κλικ γίνει στο πάνω μισό του παραθύρου,
                // το μέγεθος θα αυξηθεί κατά ένα
                if (y < 200)
                        size++;
                // αν το κλικ γίνει στο κάτω μισό του παραθύρου,
                // το μέγεθος θα μειωθεί κατά ένα
                if (y > 200)
                        size- -;
                repaint();      // επανασχεδίαση της οθόνης
                return (true);
        }

        public void paint(Graphics g) {
                // χρήση συγκεκριμένης γραμματοσειράς
                g.setFont(new Font("Helvetica", Font.BOLD, size));
                // εμφάνιση του κειμένου με το νέο μέγεθος
                g.drawString("Το μέγεθος είναι : "+size, 0, 200);
        }
}
```

Πρόγραμμα 43 – Γραφικά με Applet

Το παρακάτω applet εμφανίζει 200 αστερίσκους σε τυχαίες θέσεις στην οθόνη και σε τυχαία χρώματα και στη συνέχεια ένα όνομα από πάνω. Το όνομα θα πρέπει να μεταβιβαστεί ως παράμετρος από τη σελίδα HTML και το μέγεθος του παραθύρου να είναι 500 X 400.

```
import java.awt.*;
import java.applet.*;

public class Program43 extends Applet {
        String name;            // δήλωση String

        public void init() {
                // διάβασμα τιμής παραμέτρου από την ιστοσελίδα
                // και καταχώρησή της στη μεταβλητή name
                name = getParameter("myname");
        }

        public void paint(Graphics g) {
                // χρήση συγκεκριμένης γραμματοσειράς
                g.setFont(new Font("Helvetica", Font.BOLD, 48));
                for (int loop = 0; loop < 200; loop++) {
                        // δημιουργία τυχαίων συντεταγμένων x και y
                        int x = (int) (java.lang.Math.random()*500);
                        int y = (int) (java.lang.Math.random()*400);
                        // δημιουργία τυχαίων τιμών για τα τρία βασικά
                        // χρώματα, red, green και blue
```

```
                int red = (int) (java.lang.Math.random()*255);
                int green = (int) (java.lang.Math.random()*255);
                int blue = (int) (java.lang.Math.random()*255);
                // χρήση του χρώματος που προέκυψε από τις
                // τυχαίες τιμές των τριών βασικών χρωμάτων
                g.setColor(new Color(red, green, blue));
                // εμφάνιση ενός αστεριού σε τυχαία θέση και με
                // τυχαίο χρώμα
                g.drawString("*", x, y);
            } // τέλος του for
            // εμφάνιση του ονόματος
            g.drawString(name, 50, 200);
        }
    }
```

Ο κώδικας της ιστοσελίδας που θα καλεί το παραπάνω applet είναι ο εξής :

```
<html>
<head>
    <title>Σελίδα Εμφάνισης Applet</title>
</head>
<body>
    <p>΄Εξοδος Applet :</p>
    <p><applet code="Program43.class" width=500 height=400>
        <param name=myname value="Φλώρινα-2008">
    </applet></p>
</body>
</html>
```

Πρόγραμμα 44 – Εφέ Κίνησης

Το παρακάτω applet δημιουργεί ένα απλό εφέ κίνησης, δηλ. μια γραμμή που να κινείται σ' έναν κύκλο. Η Java υπολογίζει τις γωνίες σε ακτίνια και 2π ακτίνια είναι ίσα με 360 μοίρες. Οι συντεταγμένες του κέντρου του κύκλου είναι 120, 120 και η ακτίνα του κύκλου είναι ίση με 100.

```
import java.awt.*;
import java.applet.*;

public class Program44 extends Applet {
    // δήλωση δύο μεταβλητών για άσπρο και μαύρο χρώμα
    Color white = new Color(180, 255, 255);
    Color black = new Color(0, 0, 0);

    public void paint(Graphics g) {
        int x, y;
        x=y=0;
        double angle = 0;
```

```
                // 6,28 ακτίνια = 360 μοίρες
                while (angle < 6.28)  {
                        // υπολογισμός του συνημιτόνου της γωνίας
                        x = (int) ((java.lang.Math.cos(angle)*100)+120);
                        // υπολογισμός του ημιτόνου της γωνίας
                        y = (int) ((java.lang.Math.sin(angle)*100)+120);
                        // χρήση του μαύρου χρώματος
                        g.setColor(black);
                        // σχεδίαση γραμμής
                        g.drawLine(120,120, x, y);
                        // χρονική καθυστέρηση με for
                        for (int delay = 0; delay < 200000; delay++)
                                ;
                        // χρήση του λευκού χρώματος
                        g.setColor(white);
                        // σχεδίαση γραμμής
                        g.drawLine(120, 120, x, y);
                        angle = angle + 0.01;
                } // τέλος του while
                // χρήση του μαύρου χρώματος
                g.setColor(black);
                // σχεδίαση γραμμής
                g.drawLine(120, 120, x, y);
        }
}
```

Πρόγραμμα 45 – Περιστροφή Χαρακτήρων

Το παρακάτω applet περιστρέφει τους χαρακτήρες μιας στοιχειοσειράς, μετακινώντας κάθε χαρακτήρα προς τα αριστερά και φέρνοντας τον πρώτο στο τέλος. Η περιστροφή επαναλαμβάνεται μέχρι να φθάσουν οι χαρακτήρες στην αρχική τους θέση.

```
import java.awt.*;
import java.applet.*;

public class Program45 extends Applet {
        String text = "Florina";        // το κείμενο που θα επεξεργαστούμε
        int slength;                    // το μήκος του string
        char[ ] letter;                 // πίνακας χαρακτήρων
        // ορισμός μιας μεταβλητής γραμματοσειράς
        Font f = new Font("Serif", Font.BOLD, 24);

        public void init() {
                slength = text.length();                // μήκος του κειμένου
                letter = new char[slength];             // πίνακας χαρακτήρων
                letter = text.toCharArray();            // μετατροπή σε πίνακα
        }
```

```
        public void paint(Graphics g) {
                char temp;              // βοηθητικός χαρακτήρας
                g.setFont(f);           // εφαρμογή της γραμματοσειράς
                for (int repeat = 0; repeat < slength; repeat++) {
                        // καθάρισμα της περιοχής
                        g.clearRect(0, 0, 400, 100);
                        // εμφάνιση όλων των χαρακτήρων του πίνακα
                        g.drawChars(letter, 0, slength, 0, 50);
                        // αποθήκευση του πρώτου χαρακτήρα του πίνακα
                        temp = letter[0];
                        // ολίσθηση των χαρακτήρων του πίνακα
                        // προς τα αριστερά
                        for (int loop = 0; loop < slength-1; loop++) {
                                letter[loop] = letter[loop+1];
                                // χρονική καθυστέρηση με for
                                for (int delay = 0; delay < 1000; delay++)
                                        ;
                        } // τέλος του for
                        // τοποθέτηση του πρώτου χαρακτήρα του
                        // πίνακα στην τελευταία θέση (6)
                        letter[slength-1] = temp;
                } // τέλος του αρχικού for
        }
}
```

Η μέθοδος drawChars(), που εμφανίζει έναν ή περισσότερους χαρακτήρες ενός πίνακα, παίρνει σαν παραμέτρους το όνομα του πίνακα χαρακτήρων, τη θέση του πρώτου χαρακτήρα, τον αριθμό των χαρακτήρων και τις συντεταγμένες x, y του σημείου τοποθέτησης.

ΠΕΡΙΕΧΟΜΕΝΑ