5 Ανάπτυξη διεπαφής χρήστη

5.1 Διεπαφή χρήστη

- Η διεπαφή χρήστη (User Interface, UI) υλοποιεί την αμφίδρομη επικοινωνία συστήματος - χρήστη (υπολογιστή - ανθρώπου).
- Παραδοσιακά, σε εφαρμογές επεξεργασίας δεδομένων ο κύκλος λειτουργίας του προγράμματος ήταν: είσοδος – επεξεργασία - έξοδος και το μεγαλύτερο τμήμα του κώδικα αφιερωνόταν στην επεξεργασία. Η είσοδος γινόταν αποκλειστικά από το πληκτρολόγιο. Ουσιαστικά ο προγραμματιστής επινοούσε μια «γλώσσα» εισόδου που έπρεπε να κατέχει ο χρήστης και την οποία αποδεχόταν το σύστημα, ώστε να την επεξεργάζεται και να παράγει το αποτέλεσμα στη γλώσσα εξόδου που ήταν συνήθως γραμμογραφημένες εκτυπώσεις. Η διεπαφή χρήστη σε τέτοια συστήματα ήταν ρυθμού χαρακτήρων (character-mode UI).
- Στα πιο μοντέρνα προγράμματα για λόγους ευχρηστίας η διεπαφή βασίζεται σε γραφικά (Graphical User Interfaces, GUI). Η υλοποίηση GUI απαιτεί περισσότερο κώδικα και η εκτέλεσή τους καταναλώνει περισσότερους πόρους. Σε αρκετές περιπτώσεις συμβαίνει το πρόγραμμα να περνάει περισσότερο χρόνο σε κώδικα που υλοποιεί τη διεπαφή, παρά σε «παραγωγικό» κώδικα που υπολογίζει κάποιο αποτέλεσμα. Οι ενέργειες του χρήστη καταδεικνύονται, δεν πληκτρολογούνται. Τα δεδομένα πληκτρολογούνται, όταν αυτό είναι πιο πρακτικό από να επιλέγονται.
  - Για εξειδικευμένες εφαρμογές και έμπειρους χρήστες η χρήση GUI μπορεί να επιβραδύνει την χρήση της εφαρμογής. Γι αυτό οι εφαρμογές ρυθμού χαρακτήρων εξακολουθούν να χρησιμοποιούνται ακόμα και σήμερα, πχ στο σύστημα κρατήσεων αεροπορικών εισιτηρίων, και αλλού.

5.2 Μεθοδολογία οπτικού προγραμματισμού

- Η ανάπτυξη ξεκινά με τη σχεδίαση των φορμών, την τοποθέτηση των χειριστηρίων (controls) και τη ρύθμιση των ιδιοτήτων τους. Το αρχικό αυτό πρωτότυπο αποτελείται από dummy screens (χαζές οθόνες) και δεν υλοποιεί καμιά λειτουργικότητα παρά μόνον τη διεπαφή χρήστη. Το πρωτότυπο εξελίσσεται ως προς την αισθητική και την ευχρηστία και έπειτα προστίθεται κώδικας πίσω από τις οθόνες που προσδίδει συμπεριφορά στα διάφορα χειριστήρια που είναι τοποθετημένα πάνω τους.
- Στη Java η διαδικασία είναι: (α) κατασκευάζουμε έναν υποδοχέα (container) πάνω στον οποίο θα τοποθετηθούν τα διάφορα χειριστήρια, (β) κατασκευάζουμε τα χειριστήρια, (γ) τα τοποθετούμε πάνω στον υποδοχέα.
- Ο υποδοχέας μπορεί να είναι ένα στιγμιότυπο των κλάσεων Panel ή Frame. Ένα σημαντικό χαρακτηριστικό του υποδοχέα είναι ο **διαχειριστής διάταξης** (layout manager) που ελέγχει την τοποθέτηση (διάταξη, μέγεθος) των χειριστηρίων επί του υποδοχέα. Ορίζουμε διαχειριστή διάταξης θέτοντας την ιδιότητα `layout` του υποδοχέα σε:
  - `null` Αφήνει τα χειριστήρια στο σημείο και τις διαστάσεις που έχει επιλέξει ο προγραμματιστής. Η τοποθέτησή τους δεν αλλάζει ακόμα κι αν αλλάξουμε το μέγεθος του υποδοχέα.
  - `flowLayout` Τοποθετεί χειριστήρια (συνήθως πλήκτρα) σε αράδες από αριστερά προς τα δεξιά και έπειτα πάνω προς κάτω.
  - `BorderLayout` Τοποθετεί χειριστήρια κολλητά στα όρια του υποδοχέα (βόρεια, νότια, ανατολικά, δυτικά), έτσι ώστε η υπόλοιπη περιοχή (κέντρο) να μπορεί να καταληφθεί από το κύριο χειριστήριο, πχ μια `textArea`.
  - Υπάρχουν και πιο σύνθετοι διαχειριστές διάταξης, όπως ο `GridBagLayout`, αλλά πιο σημαντικό είναι ότι μπορούμε να τοποθετήσουμε υποδοχείς μέσα σε άλλους υποδοχείς με διαφορετικούς διαχειριστές διάταξης.

```
import java.awt.*;          import java.awt.event.*;
import javax.swing.*;       import java.text.DecimalFormat;

public class SomeGUI extends JApplet {
  JLabel lblNumber; JTextField txtNumber;
  JLabel lblResult; JTextField txtResult;
  JButton btnCompute;

  public void init() {
    Container cont = getContentPane();
    cont.setLayout(new FlowLayout());
    lblNumber = new JLabel("Number:");
    cont.add(lblNumber);
    txtNumber = new JTextField(10);
    cont.add(txtNumber);
    lblResult = new JLabel("Result:");
    cont.add(lblResult);
    txtResult = new JTextField(10);
    txtResult.setEditable(false);
    cont.add(txtResult);
    btnCompute = new JButton("Compute");
    btnCompute.addActionListener(this);
    cont.add(btnCompute);
  }

  public void actionPerformed(ActionEvent actionEvent) {
    DecimalFormat aFormat = new DecimalFormat("#.000000");
    double dblResult = Math.sqrt(Double.parseDouble(
                                    txtNumber.getText()));
    txtResult.setText(
                aFormat.format(dblResult). toString());
  }
} //end class
```

Εικόνα 62 Διεπαφή με γραφικά αντικείμενα

- Τα πιο συχνά χρησιμοποιούμενα χειριστήρια είναι τα πλήκτρα (buttons), οι ετικέτες (labels), τα πεδία κειμένου (text fields). Άλλα χειριστήρια είναι οι λίστες (lists), τα πλαίσια ελέγχου (checkboxes) και οι περιοχές κειμένου (text areas).
  - Η Java έχει δύο βιβλιοθήκες για την κατασκευή διεπαφών χρήστη: την παλιότερη AWT και τη νεότερη Swing. Η λειτουργικότητα και ο τρόπος χρήσης τους είναι αρκετά κοντινός, ωστόσο το Swing έχει περισσότερες δυνατότητες και παράγει διεπαφές με μεγαλύτερη μεταφερσιμότητα σε διάφορες υπολογιστικές πλατφόρμες.

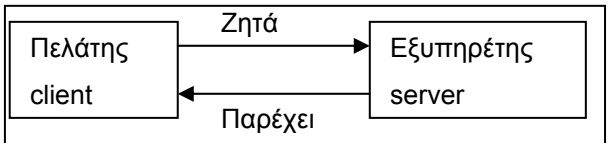
5.3 Οδηγούμενος από συμβάντα (event-driven) προγραμματισμός

- Το **συμβάν** είναι μια ενέργεια που αναγνωρίζεται και προκαλεί αντίδραση. Συνήθως οι ενέργειες προκαλούνται από την αλληλεπίδραση του χρήστη με τη διεπαφή γραφικών. Το σύστημα αντιλαμβάνεται το συμβάν. Υπάρχουν πολλά συμβάντα που λαμβάνουν χώρα, πχ η μετακίνηση δείκτη ποντικιού, το πάτημα πλήκτρου, η επιλογή με ποντίκι, ή απλώς η υπέρβαση ενός χρονικού ορίου αδράνειας. Ο προγραμματιστής ορίζει σε ποια συμβάντα να αντιδράσει το σύστημα και με ποιο τρόπο. Ο τρόπος είναι η ενεργοποίηση μιας μεθόδου χειρισμού συμβάντος (event handling method).

5.4 Χρηστικότητα διεπαφής

- Τα κυριότερα χαρακτηριστικά που καθορίζουν τη χρηστικότητα (usability) μιας διεπαφής, είναι:
  - Συνέπεια
  - Απλότητα
  - Χρήση μεταφορών από την πραγματικότητα που μοντελοποιείται από το πρόγραμμα
  - Ελαχιστοποίηση ενεργειών του χρήστη
  - Άμεση ανάδραση
  - Βοήθεια: Συμβουλές, μηνύματα, online βοήθεια ευαίσθητη στα συμφραζόμενα (context-sensitive)
  - Ελαχιστοποίηση απομνημόνευσης
  - Κατανοητά μηνύματα λαθών: εξειδικευμένα, καθοδηγητικά, με θετικό τόνο, συνεπή μορφοποίηση
  - Εναρμόνιση με προσλαμβάνουσες παραστάσεις χρήστη
  - Ευκαμψία και προσαρμοστικότητα

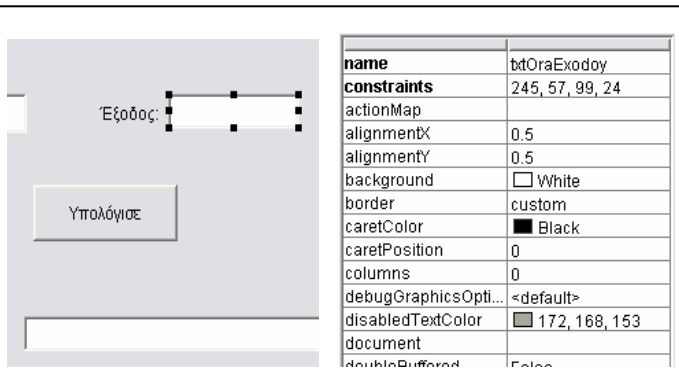
- Η διεπαφή χρήστη δεν πρέπει να συσχετίζεται με την υλοποίηση της λειτουργικότητας. Τα δύο αυτά υποσυστήματα πρέπει να επικοινωνούν μέσω της δημόσιας διεπαφής (public interface) του δεύτερου. Έτσι είναι εύκολο να υλοποιηθούν διαφορετικές διεπαφές που να επιτρέπουν στους χρήστες να προσπελαύνουν τις λειτουργίες με διαφορετικό τρόπο.
  - Μοντέλο πελάτη – εξυπηρέτη (client – server). Ένα άρθρωμα υλοποιεί συγκεκριμένη λειτουργικότητα και την προσφέρει ως υπηρεσία σε καταναλωτές (έναν ή περισσότερους). Χρησιμοποιείται ιδιαίτερα όταν ο πελάτης και ο εξυπηρέτης δεν βρίσκονται στην ίδια μηχανή, πχ στην περίπτωση κατανομής (distribution), όπως για παράδειγμα σε ένα σύστημα διαχείρισης βάσεων δεδομένων.



Εικόνα 63 Το μοντέλο πελάτη - εξυπηρέτη

5.5 Χειριστήρια (Controls ή components)

- Τα χειριστήρια είναι επαναχρησιμοποιούμενα στοιχεία της διεπαφής χρήστη με τυποποιημένη, αλλά προσαρμοζόμενη συμπεριφορά.
  - Η συμπεριφορά τους αλλάζει μέσω των ιδιοτήτων (properties) που μπορούν να τεθούν στις αρχικές τους τιμές πάνω στον πίνακα ιδιοτήτων, ενώ αρκετές μπορούν να αλλάζουν και κατά την διάρκεια εκτέλεσης του προγράμματος (runtime).
  - Η παλαιότερη βιβλιοθήκη AWT υλοποιεί στην κλάση `java.awt.Component` τα χειριστήρια, όπως τα `Button`, `Label`, `TextField`, `Checkbox`.

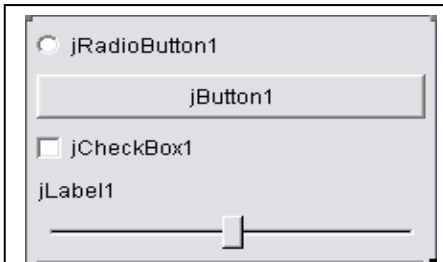


Εικόνα 64 Χειρισμός ιδιοτήτων στον JBuilder

- Εκτός από τα βασικά χειριστήρια, όπως πλαίσιο κειμένου, ετικέτα, πλήκτρο εντολής, λίστα επιλογών, σύνθετο πλαίσιο, που παρέχονται από το περιβάλλον προγραμματισμού, υπάρχουν και πρόσθετα χειριστήρια που διατίθενται με τη μορφή κλάσεων / βιβλιοθηκών και χρησιμοποιούνται όποτε χρειάζονται. Μερικές φορές τα χειριστήρια αναπτύσσονται από εταιρία που πωλεί το δικαίωμα χρήσης.

- Υπάρχουν χειριστήρια που διαχειρίζονται εικόνες, προβάλλουν γραφικά αντικείμενα (γραμμές, ελλείψεις, τετράπλευρα, κά) και λειτουργούν όπως τυποποιημένα αντικείμενα γραφικών περιβαλλόντων (γραμμές κύλισης, πλαίσια διαλόγου καταλόγων και αρχείων, μενού).

- Κάθε χειριστήριο έχει συνδεθεί με ένα σύνολο συμβάντων, πχ με το πάτημα του ποντικιού, με την εστίαση, με την απώλεια εστίασης. Όταν λάβει χώρα το συμβάν, εκτελείται ο κώδικας που έχει γραφτεί γι αυτό.



5.6 Ασκήσεις

Α53. Κατασκευάστε ένα πρόγραμμα που θα απεικονίζει τους φωτεινούς σηματοδότες μιας διασταύρωσης. Κάθε κύκλος αλλαγής της ροής της πληροφορίας θα διαρκεί συγκεκριμένο χρονικό διάστημα, πχ 1 λεπτό.

Α54. Χρησιμοποιήστε τις κλάσεις ουράς και στοίβας και υλοποιήστε μια διεπαφή (interface) που να απεικονίζει οπτικά τη χρήση τους.

Α55. Υλοποιήστε προσομοιωτή φορητού ραδιοφώνου κατασκευάζοντας τηλεχειριστήριο και οθόνη ενδείξεων.

Α56. Παρομοίως υλοποιήστε διεπαφή για τηλεοπτικό δέκτη με On Screen Display.

Α57. Παρομοίως υλοποιήστε διεπαφή φορητού CD player.

Α58. Υλοποιήστε διεπαφή μιας απλής φωτογραφικής μηχανής.

Α59. Υλοποιήστε διεπαφή φούρνου μικροκυμάτων.

Α60. Υλοποιήστε ρυθμιστικό τηλεχειριστήριο κλιματιστικού.

Α61. Υλοποιήστε χειριστήρια και πίνακα ενδείξεων ταχυπλόου.

Α62. Υλοποιήστε διεπαφή για επιλογή τραγουδιών από συσκευή που περιέχει πλήθος τραγουδιών (τζουκ μποξ).

Α63. Υλοποιήστε διεπαφή για δημοσιογραφικό κασετόφωνο.

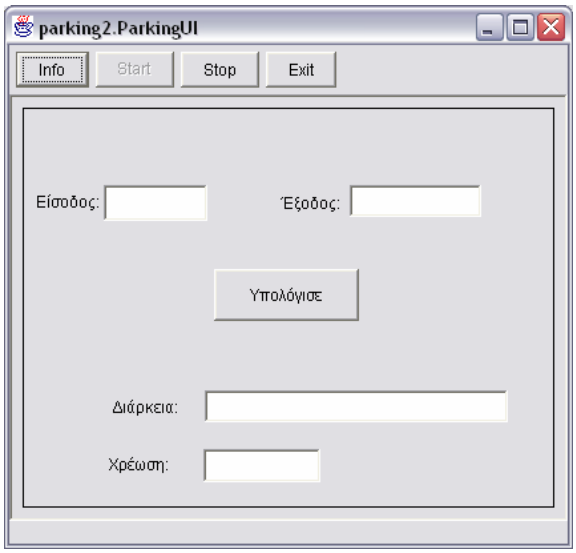
Α64. Υλοποιήστε αριθμομηχανή με ταινία χαρτιού (να μετατρέπει από δραχμές σε ευρώ και αντίστροφα).

Α65. Φτιάξτε πρόγραμμα που παρακολουθεί την κίνηση των πελατών μιας τράπεζας με τρία ταμεία.

5.7 Ώρα και πάρκινγκ με γραφική διεπαφή

Ακολουθεί μια εναλλακτική υλοποίηση της άσκησης Α42, σελ. 46. Η άσκηση αυτή έχει υλοποιηθεί άλλη μια φορά με το πρόγραμμα της ενότητας 3.8, σελ. 47 που έχει δύο κλάσεις: την κλάση Ora που διαχειρίζεται αντικείμενα τύπου ώρας, και την κλάση Parking που υλοποιεί μια διεπαφή χρήστη που χρησιμοποιεί παράθυρα διαλόγου για να διαβάσει τις εισόδους του προγράμματος. Εδώ παρατίθεται μια κλάση ParkingUI υλοποιεί μια Swing-based διεπαφή που έχει κατασκευαστεί από το design tool του JBuilder. Για να μεταγλωττιστεί το πακέτο απαιτείται να συμπεριληφθεί και η κλάση Ora. Το μεγαλύτερο τμήμα του κώδικα παρήχθη αυτόματα με βάση τη σχεδίαση που φαίνεται στην Εικόνα 66 και έπειτα απλοποιήθηκε (πχ αφαιρέθηκε ο λεπτομερειακός χειρισμός εξαιρέσεων). Με κάθετη γραμμή στο αριστερό περιθώριο σημειώνεται ο κώδικας που γράψαμε εμείς και υλοποιεί τη λειτουργικότητα (όχι τη διεπαφή) του προγράμματος.

```
package parking2;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;

public class ParkingUI extends JApplet {

  final static double MINIMUM_TIMH = 2.0;
  final static double MINIMUM_ORA  = 3.0;
  final static double TIMH_ANA_ORA = 0.5;
  final static double MAXIMUM_TIMH = 6.0;

  private JPanel jPanel1 = new JPanel();
  private XYLayout xYLayout1 = new XYLayout();
  private JTextField txtOraEisodoy = new JTextField();
  private JTextField txtOraExodoy = new JTextField();
  private JButton jButton1 = new JButton();
  private JTextField txtDiarkeia = new JTextField();
  private JTextField txtXreosi = new JTextField();
  private JLabel jLabel1 = new JLabel();
  private JLabel jLabel2 = new JLabel();
  private JLabel jLabel3 = new JLabel();
  private JLabel jLabel4 = new JLabel();

    public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  //Component initialization
  private void jbInit() throws Exception {
```



Εικόνα 66 Γραφική διεπαφή με το Swing

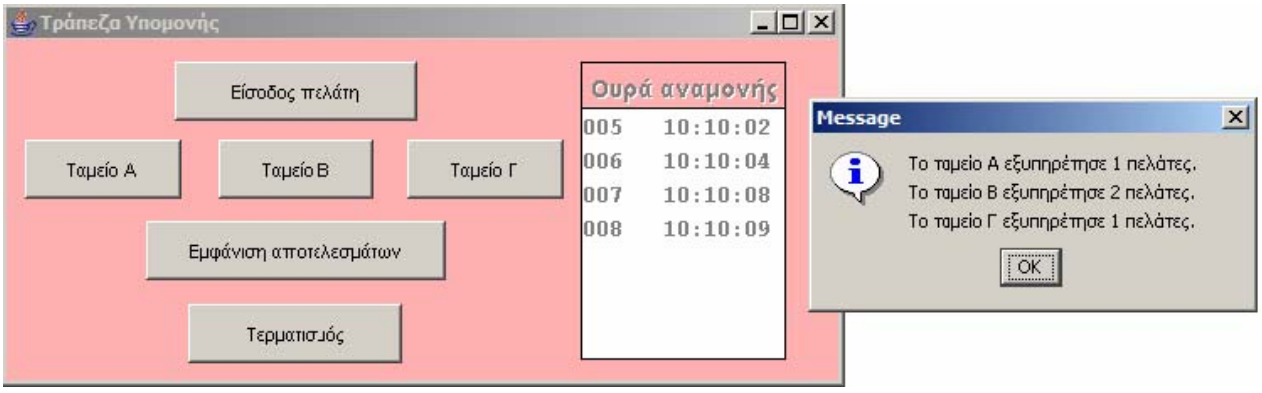
```
    this.setSize(new Dimension(400,300));
    jPanel1.setLayout(xYLayout1);
    jButton1.setText("Υπολόγισε");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });
    txtDiarkeia.setEnabled(false);
    txtDiarkeia.setEditable(false);
    txtXreosi.setEditable(false);
    txtXreosi.setEnabled(false);
    jLabel1.setText("Είσοδος:");
    jLabel2.setText("Έξοδος:");
    jLabel3.setText("Διάρκεια:");
    jLabel4.setText("Χρέωση:");
    this.getContentPane().add(jPanel1, BorderLayout.CENTER);
    jPanel1.add(jLabel1, new XYConstraints(9, 58, 88, 22));
    jPanel1.add(txtOraEisodoy,    new XYConstraints(60, 57, 78, 27));
    jPanel1.add(txtOraExodoy,  new XYConstraints(245, 57, 99, 24));
    jPanel1.add(jLabel2, new XYConstraints(193, 59, 88, 22));
    jPanel1.add(jButton1, new XYConstraints(143, 120, 109, 39));
    jPanel1.add(jLabel3, new XYConstraints(66, 212, 88, 22));
    jPanel1.add(txtDiarkeia,   new XYConstraints(136, 211, 228, 24));
    jPanel1.add(jLabel4, new XYConstraints(64, 255, 88, 22));
    jPanel1.add(txtXreosi,   new XYConstraints(135, 255, 88, 25));
  }

  private static double computeDiafora(Ora ora1, Ora ora2) {
    return ora2.convertOra() - ora1.convertOra();
  }

  private static double computeTimh(double diarkeia) {
    double dTimh;
    dTimh = MINIMUM_TIMH;
    if (diarkeia > MINIMUM_ORA) {
      diarkeia -= MINIMUM_ORA;
      dTimh += TIMH_ANA_ORA * Math.ceil(diarkeia);
    }
    if (dTimh > MAXIMUM_TIMH) dTimh = MAXIMUM_TIMH;
    return dTimh;
  }

  void jButton1_actionPerformed(ActionEvent e) {
    Ora eisodos = new Ora(txtOraEisodoy.getText());
    Ora exodos = new Ora(txtOraExodoy.getText());
    double diarkeia = computeDiafora(eisodos, exodos);
    txtDiarkeia.setText(String.valueOf(diarkeia));
    double timh = computeTimh(diarkeia);
    txtXreosi.setText(String.valueOf(timh));
  }
}
```

5.8 Τράπεζα (ουρά στα ταμεία)

[Εργασία ακαδ. έτους 2001] Η «Ωμέγα Τράπεζα Υπομονής» έχει δεχθεί πολλά παράπονα από τους πελάτες ενός υποκαταστήματός της για μεγάλες καθυστερήσεις στην εξυπηρέτησή τους. Επίσης, κατά καιρούς οι υπάλληλοι διαμαρτύρονται ότι άλλοι συνάδελφοί τους δεν εργάζονται τόσο εντατικά όσο οι ίδιοι. Ο διοικητής της τράπεζας λοιπόν σας ανέθεσε να διερευνήσετε τι πραγματικά συμβαίνει.

Το σύστημα εξυπηρέτησης πελατών λειτουργεί ως εξής. Με την είσοδό τους στην τράπεζα οι πελάτες παίρνουν ένα χαρτάκι με τον αύξοντα αριθμό τους και κάθονται στο χώρο αναμονής. Το υποκατάστημα έχει τρία ταμεία, τα Α, Β και Γ στα οποία προσέρχονται οι πελάτες για να εξυπηρετηθούν, όταν τους καλέσει ο ταμίας με τον αριθμό τους.

Φτιάξτε ένα πρόγραμμα που να παρακολουθεί την αναμονή των πελατών και να υπολογίζει την παραγωγικότητα των ταμείων. Φανταστείτε ότι ο χρήστης του προγράμματος θα είναι κάπου κρυμμένος μέσα στην τράπεζα και θα καταγράφει πότε μπήκε κάθε πελάτης και πότε και από ποιο ταμείο εξυπηρετήθηκε. Στο τέλος της "ημέρας" θα ανακοινώνονται στο διευθυντή τα αποτελέσματα της έρευνας.



Εικόνα 67 Δείγμα GUI για τράπεζα

Μια πρόταση για τη διεπαφή χρήστη είναι μια επιφάνεια όπως φαίνεται στην Εικόνα 67. Όταν μπαίνει ένας πελάτης στην ουρά αναμονής, ο χρήστης πατάει το «Είσοδος πελάτη» και στον παράπλευρο πίνακα «Ουρά αναμονής» καταγράφεται ο αριθμός του πελάτη και η ώρα που μπήκε στην τράπεζα. Όταν κάποιος ταμίας καλεί τον επόμενο πελάτη από την ουρά, ο χρήστης πατάει το αντίστοιχο πλήκτρο «Ταμείο Χ», όπου Χ είναι Α, Β, Γ. Ο παλιότερος πελάτης (με τον μικρότερο αριθμό) φεύγει από την ουρά αναμονής και εξυπηρετείται από τον ταμία που τον κάλεσε.

Το πλήκτρο «Εμφάνιση αποτελεσμάτων» εμφανίζει το πλήθος των πελατών που εξυπηρέτησε ο κάθε ταμίας (για να δει ο διευθυντής ποιος είναι ο πιο παραγωγικός).. **Προαιρετικά:** θα ήταν χρήσιμο να έδινε κι άλλα στοιχεία, όπως τον μέσο χρόνο αναμονής των πελατών (για να δει ο διευθυντής αν έχουν δίκιο να παραπονιούνται οι πελάτες του υποκαταστήματος), τον μέγιστο χρόνο αναμονής πελάτη, κά.

Για να φτιάξετε το πρόγραμμα **υποχρεωτικά** θα χρησιμοποιήσετε τον παρακάτω κώδικα που υλοποιεί μια **ουρά** χρησιμοποιώντας ένα διάνυσμα. Θεωρήστε ότι ανά πάσα στιγμή το σαλόνι της τράπεζας μπορεί να χωρέσει το πολύ 7 άτομα (= `Q_LENGTH`). Παρατηρήστε ότι τα στοιχεία της ουράς είναι συμβολοσειρές. Κατά σύμβαση στις πρώτες 3 θέσεις της συμβολοσειράς θα τοποθετείτε τον αριθμό του πελάτη και στις 8 επόμενες την ώρα εισόδου, πχ ο πελάτης με αριθμό 3 θα τοποθετηθεί στην ουρά ως **`003 12:31:23`**.

Είναι καλοδεχούμενο να σχεδιάσετε μια **διαφορετική διεπαφή χρήστη**. Μάλιστα μπορείτε να υποστηρίξετε ότι η διεπαφή του προγράμματος αυτού δεν χρειάζεται να είναι ρυθμού γραφικών (Graphical User Interface, GUI). Για παράδειγμα, μια "παλιομοδίτικη" αλλά εξίσου, ή ίσως και πιο εύχρηστη, **διεπαφή ρυθμού χαρακτήρων** (character-based UI): όταν μπαίνει ένας πελάτης ο χρήστης θα πατάει "Ε", όταν εξυπηρετείται πελάτης από το ταμείο Α θα πατάει "Α", κοκ.

Ακολουθεί ο κώδικας που υλοποιεί μια ουρά. Αντιγράψτε τον όπως είναι στο αρχείο “ArrayQueue.java”.

Επέκταση 1: (α) Γενικεύσετε το πρόγραμμα έτσι ώστε να δέχεται ως παράμετρο τον αριθμό των ταμείων. (β) Όλα τα ταμεία δεν είναι συνέχεια διαθέσιμα: το πρωί δεν ξεκινούν όλα τα ταμεία, καθώς δεν υπάρχουν πολλοί πελάτες, και οι ταμίες σταματούν για διάλειμμα και ξεκινούν παλι. Ενσωματώστε λειτουργία που θα θέτει εντός / εκτός λειτουργίας κάθε ταμείο. Στα αποτελέσματα θα περιλαμβάνονται ο χρόνος λειτουργίας του ταμείου, πόσες φορές διακόπηκε η λειτουργία του και ό,τι άλλη πληροφορία κρίνετε ότι θα βοηθούσε τον διευθυντή.

Επέκταση 2: Τροποποιήστε το πρόγραμμά σας έτσι ώστε να δέχεται είσοδο όχι με χρήση του GUI, αλλά διαβάζοντας στοιχεία από ένα αρχείο κειμένου του οποίου το format θα καθορίσετε εσείς. Τροφοδοτήστε με εικονικά στοιχεία από μια γεννήτρια τυχαίων αριθμών το αρχείο κειμένου. Τα αποτελέσματα αντί να εμφανίζονται στην οθόνη, θα αποθηκεύονται σε ένα αρχείο κειμένου. Αν έχετε σχεδιάσει καλά το πρόγραμμά σας, οι αλλαγές στον κώδικα πρέπει να είναι μικρές.

```
public class ArrayQueue {

  //μέγιστο πλήθος στοιχείων που μπορούν να τοποθετηθούν στην ουρά
  private final static int Q_LENGTH = 7;

  private String[] q = new String[ArrayQueue.Q_LENGTH + 1]; //πίνακας όπου αποθηκεύονται στοιχεία
  private int head; //δείχνει τη θέση του πρώτου στοιχείου της ουράς
```

```
private int tail; //δείχνει τη θέση στην οποία θα μπει το επόμενο στοιχείο

/**
  * ArrayQueue Κατασκευαστής κενής ουράς
  */
public ArrayQueue() {this.head = 0; this.tail = 0; } //κατασκευαστής

/**
 * isEmpty Ελέγχει αν η ουρά είναι άδεια
 * @return boolean  Αληθές αν η ουρά είναι άδεια, ψευδές διαφορετικά
 */
public boolean isEmpty() { return this.head == this.tail; }

/**
 * isFull Ελέγχει αν η ουρά είναι γεμάτη
 * @return boolean Αληθές αν η ουρά είναι γεμάτη, ψευδές διαφορετικά
 */
public boolean isFull() { return ArrayQueue.incr(this.tail) == this.head; }

/**
 * enQueue Τοποθετεί ένα στοιχείο στην ουρά
 * @param item String Το στοιχείο που τοποθετείται στην ουρά
 * @throws Exception Εγειρεται όταν η ουρά είναι γεμάτη και δεν έχει χώρο να εισάγει νέο στοιχείο
 */
public void enQueue(String item) throws Exception {
  if (this.isFull())
    throw new Exception("Ουρά γεμάτη - Το στοιχείο δεν μπορεί να εισαχθεί.");
  else {
    q[this.tail] = item;  //τοποθέτησε στην ουρά
    this.tail = ArrayQueue.incr(this.tail); //ενημέρωσε τον δείκτη τέλους
  }
} //end enQueue

/**
 * deQueue Αφαιρεί το παλιότερο στοιχείο από την ουρά
 * @return String Αν η ουρά δεν είναι άδεια, αφαιρεί το παλιότερο στοιχειο.
 * @throws Exception Εγείρεται όταν η ουρά ειναι άδεια, οπότε δεν υπάρχει στοιχείο να επιστραφεί
 */
public String deQueue() throws Exception {
  if (this.isEmpty())
    throw new Exception("Ουρά άδεια - Δεν επιστρέφεται στοιχείο.");
  else {
    String temp = q[this.head]; //πάρε το στοιχείο
    this.head = incr(this.head); //ενημέρωσε τον δείκτη κεφαλής
    return temp; //επέστρεψε
  }
}

/**
 * toString Προβάλλει τα περιεχόμενα της ουράς
 * @return String Εμφανίζει κάθε στοιχειο και τη θέση του στην ουρά
 */
public String toString() {
  int i = this.head;
  String s = "";
  while (i != this.tail) {
    s += (i + ":" + q[i] + " ");
    i = ArrayQueue.incr(i);
  }
  return s;
}

private static int incr(int i) {  //βοηθητική μέθοδος
  return (i + 1) % (ArrayQueue.Q_LENGTH + 1); }

public static void main(String[] args) {
  ArrayQueue a = new ArrayQueue();
  try {
    a.enQueue("ασταρι"); a.enQueue("στοκος"); a.enQueue("μπογια");
    System.out.println(a.toString());
    a.deQueue(); a.deQueue(); a.deQueue();
  }
  catch (Exception ex) { System.out.println(ex); }
  System.exit(0);
```

```
  } //end main
} //end class
```

Ακολουθούν τμήματα του κώδικα που υλοποιεί τη γραφική διεπαφή χρήστη. Ο προγραμματιστής φτιάχνει το παρουσιαστικό με κάποιο εργαλείο σχεδίασης GUI και ακολούθως προσθέτει τον παρακάτω κώδικα σε κάθε πλήκτρο.

```
package bankqueue;

//εδώ υπάρχουν εντολές import

public class BankFrame extends JFrame {

  /// ------------------------------------------------------------------------
  /// οι δηλώσεις
  private ArrayQueue q = new ArrayQueue(); /// φτιάχνει ένα αντικείμενο Ουρά
  DecimalFormat counterFormat = new DecimalFormat("000"); /// τριψήφιος αριθμός πελάτη
  SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm:ss");  /// το φορμά τη ημερομηνίας
  private int pelates  = 0; /// μετράει τους πελάτες που μπαίνουν στην τράπεζα
  private int pelatesA = 0; /// μετράει τους πελάτες που έχει εξυπηρετήσει το ταμείο Α
  private int pelatesB = 0; /// μετράει τους πελάτες που έχει εξυπηρετήσει το ταμείο Β
  private int pelatesC = 0; /// μετράει τους πελάτες που έχει εξυπηρετήσει το ταμείο Γ
/// ο κώδικας που εκτελείται όταν πατιούνται τα κουμπιά

  void jButtonEisodos_mouseClicked(MouseEvent e) { //πλήκτρο «Είσοδος πελάτη»
    if (q.isFull())
      JOptionPane.showMessageDialog(null,
        "Η αίθουσα αναμονής της τράπεζας είναι γεμάτη. Παρακαλώ περάστε αργότερα.");
    else
      try {
        pelates ++; /// αύξησε το πλήθος των πελατών που μπήκαν μέσα στην τράπεζα
        /// φτιάξε την γραμμή που περιέχει αύξοντα αριθμό του πελάτη και ώρα εισόδου στην τράπεζα
        String s = counterFormat.format(pelates) + "   " + timeFormat.format(new Date());
        q.enQueue(s); /// τοποθέτησε τα στοιχεία στην ουρά
        jta.append(s + "\n"); /// εμφάνισε τα στοιχεία στον πίνακα ανακοινώσεων
      }
      catch (Exception ex) { System.out.println(e); }
  }

  void jButtonTameioA_mouseClicked(MouseEvent e) { //πλήκτρο «Ταμείο Α» - ομοίως και στα Β, Γ
    if (q.isEmpty())
      JOptionPane.showMessageDialog(null, "Δεν υπάρχει πελάτης στην ουρά.");
    else
      try {
        pelatesA++; /// αύξησε τον μετρητή των πελατών που εξυπηρέτησε το ταμείο Α
        q.deQueue();
        jta.replaceRange("", 0, 15); /// σβήσε την πρώτη γραμμή από τον πίνακα ανακοινώσεων
      }
      catch (Exception ex) { System.out.println(ex); }
  }

    void jButtonApotelesmata_mouseClicked(MouseEvent e) { // πλήκτρο «Εμφάνιση αποτελεσμάτων»
    JOptionPane.showMessageDialog(null, /// εμφάνισε τα αποτελέσματα
      "Το ταμείο Α εξυπηρέτησε " + pelatesA + " πελάτες.\n" +
      "Το ταμείο Β εξυπηρέτησε " + pelatesB + " πελάτες.\n" +
      "Το ταμείο Γ εξυπηρέτησε " + pelatesC + " πελάτες.\n");
    }

  void jButtonTermatismos_mouseClicked(MouseEvent e) { //πλήκτρο «Τερματισμός»
    System.exit(0); /// τερμάτισε το πρόγραμμα
  }
}
```

6 Σύνταξη γλωσσών

6.1 Ιδεατά (conceptual) μοντέλα σχεδίασης

6.2 Ταχεία πρωτοτυποποίηση (rapid prototyping)

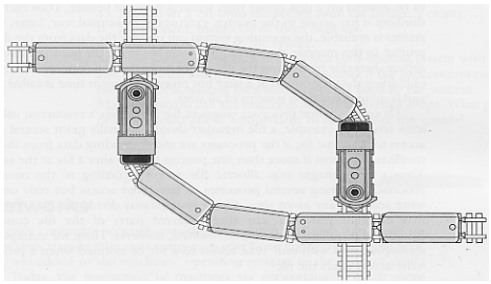
6.3 Ακραίος προγραμματισμός (extreme programming)

6.4 Πολλαπλά νήματα (multithreading)

- Σαν να έχεις πολλές διεργασίες (processes) που μοιράζονται την ίδια μνήμη
- Παράδειγμα: ασύγχρονη μεταφορά δεδομένων, επεξεργασία, προώθηση και προβολή κατ' απαίτηση στο δίκτυο

6.5 Ανταγωνισμός μεταξύ διεργασιών

- Υπάρχει ένας πόρος (resource) και οι ενεργές διεργασίες (processes) διαγωνίζονται για την αποκλειστική απόκτησή του. Η διεργασία που θα τον αποκτήσει, αποκλείει αυτόματα όλες τις άλλες από τη διεκδίκησή του, μέχρι να τον απελευθερώσει.
- Σημαφόρος (semaphore): δυαδικό ψηφίο με καταστάσεις set και clear. Μπορούμε να ελέγξουμε την κατάστασή του και να του θέσουμε τιμή χωρίς αυτή η λειτουργία να διακοπεί
  - Απενεργοποιώντας τις εξαιρέσεις (interrupts)
  - Χρησιμοποιώντας ειδική εντολή της γλώσσας μηχανής Test And Set που ολοκληρώνεται αδιάλειπτα
- Το τμήμα του κώδικα που απαιτεί αποκλειστική κατοχή ενός πόρου από μια διεργασία ονομάζεται κρίσιμη περιοχή (critical region). Ο αμοιβαίος αποκλεισμός (mutual exclusion) υλοποιείται με την θέση (set) μιας σημαφόρου στην αρχή και τον καθαρισμό της (clear) στο τέλος.
- Αδιέξοδο (deadlock) είναι η κατάσταση όπου δύο ή περισσότερες διεργασίες εμποδίζονται αμοιβαία να προχωρήσουν επειδή αναμένουν την προσπέλαση στους ίδιους πόρους.



Εικόνα 68 Αδιέξοδο στη διασταύρωση

7 Παραπομπές

7.1 Βιβλία

Στους φοιτητές διανέμεται το βιβλίο [18]. Ωστόσο, οι φοιτητές ενθαρρύνονται να ανατρέχουν κατά περίπτωση και στις άλλες αναφορές που παρατίθενται.

1. David Barnes, Michael Koelling, *Objects First with Java,* 2nd ed., Pearson, 2005, 0131249339, υλικό στο www.bluej.org
2. Bruce Eckel, *Thinking in Java,* 3rd ed., 2002, downloadable from http://www.mindview.net/Books/TIJ/
3. David Harel, *Algorithmics: the spirit of computing,* 2nd ed., Addison Wesley, 1992, 0-201-50401-4
4. Deitel & Deitel, *Java How to program,* 3rd ed., Prentice Hall, 1999, 0-13-012507-5
5. Douglas Bell, Ian Morey, John Pugh, *The essence of program design,* Prentice Hall, 1997, 0-13-367806-7
6. Ellis Horowitz, *Βασικές αρχές γλωσσών προγραμματισμού,* 2η έκδοση, Κλειδάριθμος, 1993, 960-209-190-6
7. J. Glenn Brookshear, *Computer Science, an overview,* 6th ed., Addison Wesley, 2000, 0-201-35747-X
8. Kernigham, Plauger, *The elements of programming style,* McGraw-Hill, 2nd ed, 1978
9. M. Bohl, M. Rynn, *Tools for structured design: an introduction to programming logic*, 5th ed., Prentice-Hall, 2001, 0-13-020037-9
10. Maureen Sprankle, *Problem solving and programming concepts,* 5th ed., Prentice Hall, 2001, 0-13-022967-9
11. P. H. Winston, S. Narasimhan, *On to Java 1.2,* 2nd ed., Addison Wesley, 1998, 0-201-38598-8
12. Ravi Sethi, *Programming languages,* 2nd ed., Addison Wesley, 1996, 0-201-59065-4
13. Ravi Sethi, *Programming Languages: Concepts and Constructs,* 2nd ed., Addison Wesley, 1996, 0-201-59065-4
14. Robert W. Sebesta, *Concepts of Programming Languages,* 5th ed., Addison Wesley, 2001, 0-201-75295-6
15. Roger Graham, *The synergy of hardware and software,* Prentice Hall, 1998, 0-13-145617-2
16. Stefano Ceri, Dino Mandrioli, Licia Sbattella, *The art and craft of computing,* Addison Wesley, 1998, 0-201-87698-1
17. Terry Winograd, *Bringing design to software,* ACM Press, 2000, 0-201-85491-0
18. Γιώργος Λιακέας, *Εισαγωγή στη Java 2*, Κλειδάριθμος, 2003, 960-209-625-X
19. John Hubbard, *Schaum's Java Θεωρία και προβλήματα,* Κλειδάριθμος, 2005, 960-209-737-X
20. Ι. Ανδρεάδης, Μια σύντομη αναδρομή στην τεχνολογία των ηλεκτρονικών υπολογιστών, *Δελτίο ΠΣΔΜ-Η,* σελ. 48-52, Ιούλιος - Αύγουστος 2001
21. Ι. Κάβουρα, *Δομημένος προγραμματισμός με Pascal,* Κλειδάριθμος, 1997, 960-209-308-0
22. ΥΠΕΠΘ, Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον, 1999, 960-7251-23-7
23. Sartaj Sahni, *Data Structures, algorithms and applications in Java*, McGraw-Hill, 2000, 0-07-109217-X
24. John Lewis, William Loftus, *Java software solutions: Foundations of program design,* Addison-Wesley, 1998, 0-201-57164-1
25. John Hunt, *Java and object-orientation,* Springer-Verlag, 1997, 3-540-76201-9
26. Steven Haines, *Οδηγός της Java2*, Β. Γκιούρδας, 960-387-041-2
27. Greg Perry, *Εισαγωγή στον προγραμματισμό,* Μ. Γκιούρδας, 960-512-33-9
28. John Satzinger, Tore Orvik, *The object-oriented approach,* 2nd ed., Course Technology, 2001, 0-619-03390-8

7.2 Διαδίκτυο

1. http://www.it.uom.gr/project/java/tutorial.htm Διδακτικό βοήθημα στα ελληνικά (Πανεπιστήμιο Μακεδονίας)
2. http://chortle.ccsu.ctstateu.edu/CS151/cs151java.html Άφθονο διδακτικό υλικό, κουιζ και ασκήσεις. Ευρύτητα κάλυψης.
3. http://www.faqs.org/docs/javap/index.html Ηλεκτρονικό βιβλίο για προγραμματισμό σε Java με κουιζ και ασκήσεις.
4. http://www.tipsmart.com/javacert/ptp/basics.htm Κουιζ για τα βασικά της Java
5. http://alpha.physics.uoi.gr/web_kokkas_java/java_lang.htm Εκπαιδευτικό υλικό για Java (Πανεπιστήμιο Ιωαννίνων)
6. http://users.ntua.gr/ge04060/java.htm Ιστοσελίδα μαθήματος από το Μετσόβιο Πολυτεχνείο που χρησιμοποιεί το BlueJ
7. http://www.cs.teilar.gr/gkakaron/java/Index.html Μαθήματα Java από το ΤΕΙ Λάρισας

8 Ευρετήριο ασκήσεων & πρακτικής (με αριθμό σελίδας)

9 Δεσμευμένες λέξεις της Java

Οι παρακάτω λέξεις είναι δεσμευμένες και δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών ή μεθόδων σε προγράμματα Java.

abstract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
false	final	finally	float	for
goto	if	implements	import	instanceof
int	interface	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
true	try	void	volatile	while

Ελέγξτε πόσες από αυτές αναγνωρίζετε τι κάνουν ή πού χρησιμοποιούνται. Ποιες αναφέρονται σε βασικούς τύπους δεδομένων, ποιες σχετίζονται με γλωσσικές δομές και πώς συντάσσονται; Ποια η σημασιολογία των γλωσσικών δομών; Ποιες δεσμευμένες λέξεις σχετίζονται με αντικειμενοστρεφή χαρακτηριστικά της Java;