

Limbajul de Programare: Java

Lucrare pentru atestarea
competențelor profesionale

Absolvent
Carp Florin-Cristian
Clasa a XII-a B

Coordonator
Prof. Pilat Elena Mihaela

Limbajul de Programare: Java

Lucrare pentru atestarea
competențelor profesionale

Absolvent
Carp Florin-Cristian
Clasa a XII-a B

Coordonator
Prof. Pilat Elena Mihaela

Cuprins

Capitolul I: Motivul alegerii temei.....	1
Capitolul II: Java: Informatii de baza.....	2
Capitolul III: <i>"Hello World"</i> : Prima aplicatie in Java.....	4
Capitolul IV: Folosirea instrumentelor grafice in Java.....	6
Capitolul V: Joc interactiv.....	8
Structură aplicație.....	9
Detalii tehnice de implementare.....	10
Resurse hardware și software.....	15
Modalități de utilizare.....	16
Modalități de dezvoltare.....	17
Cod sursă.....	18
Bibliografie.....	22

Motivul alegerii temei

Java este unul dintre limbajele de programare care ne oferă siguranța de stabilitate a unei aplicații, de altfel, am ales să prezint această temă deoarece m-am confruntat cu nenumărate probleme în toți acești ani când am descoperit diferite limbaje de programare, începând cu C/C++, TCL, python, perl, până la PHP, C# și chiar Java.

De fiecare dată când am avut nevoie de un ajutor pentru a verifica, spre exemplu, validitatea datelor dintr-un formular HTML am utilizat întotdeauna Java, doar pentru simplul fapt că flexibilitatea acestui limbaj de programare ne conferă nu numai siguranță, dar și un design plăcut și lipsit de bug-uri, ceea ce este un lucru destul de important în zilele noastre. Un alt exemplu când am folosit Java a fost atunci când am avut nevoie să creez o funcție care să mă ajute, spre exemplu, să fac un buton să fie animat (minimal, cel puțin – însă poate ajuta în cazuri destul de avansate), astfel compatibilitatea și fiabilitatea acestui limbaj de programare cu tehnologiile web este foarte utilă.

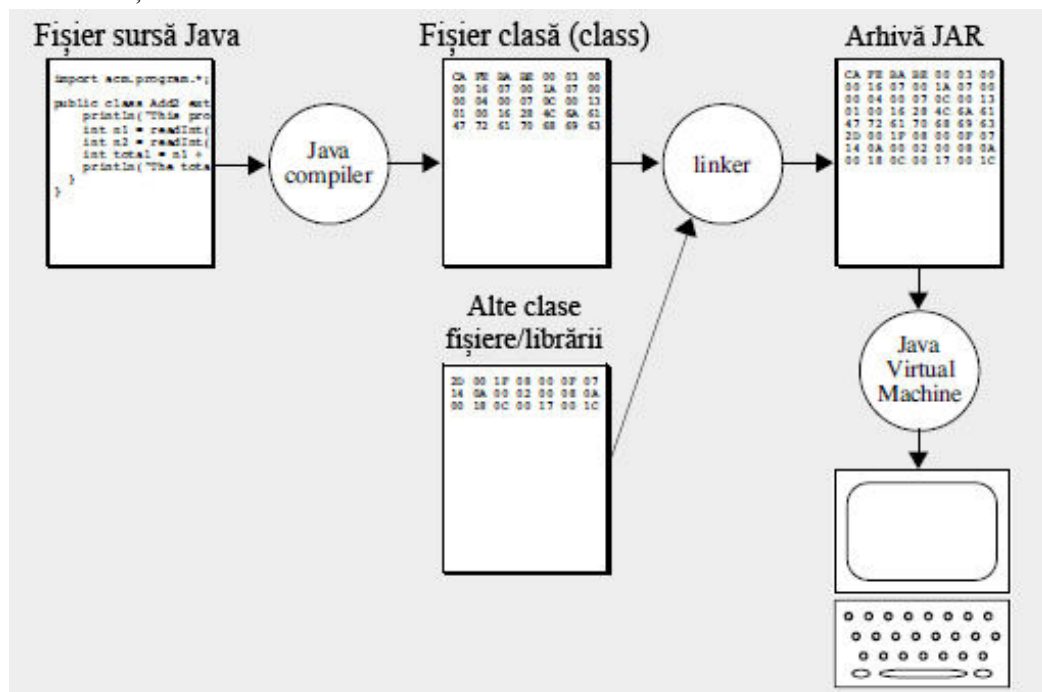
Deși acest limbaj de programare pare complex, și extrem de sofisticat la prima vedere, acesta este accesibil oricărei persoane care a studiat cel puțin un limbaj de programare precum C++ sau perl / python. Astfel, cu astfel de cunoștințe și cu o experiență minimală în webdesign și/sau webdevelopment folosind limbaje de programare adiacente precum HTML, PHP sau chiar doar cu o pasiune exacerbată pentru internet sau aplicații pentru telefon oricine are acces la o arie nelimitată de desfășurare folosind Java.

În acest mod, datorită experienței mele în domeniul web am remarcat faptul că un număr destul de mare de website-uri au incorporat cel puțin o funcție Java care fie că permite o animație minimală a unui buton, sau chiar a întregului website – sau doar pentru a realiza un client de webchat prin intermediul Java – toate acestea aducând acest limbaj de programare în topul celor mai utilizate limbaje de programare pentru soluții web optime.

II. Java: Informații de baza

Cel mai recent capitol din istoria programării orientate pe obiecte este reprezentat de dezvoltarea Java de către o echipă de programatori de la Sun Microsystems conduși de James Gosling. În 1991, când Sun a inițiat proiectul care mai târziu va deveni Java, țelul era să dezvolte o limba de programare care să fie compatibilă pentru programarea microprocesoarelor din device-urile electronice. Deși acest proiect era în desfășurare, direcția acestuia s-a schimbat brusc în momentul în care s-a observat o creștere fenomenală a Internetului în forma WWW (World Wide Web) printr-o continuă creștere a unei colecții de resurse interconectate la care contribuiau utilizatori din toată lumea. Pe când acest lucru avea loc în 1993, Sun a decis să modeleze Java ca un limbaj de programare compatibil și cu web-ul, respectiv ca o soluție pentru a dezvolta aplicații web interactive. Această decizie a fost extraordinară. Încă din Mai 1995, când Sun a anunțat realizarea acestui limbaj de programare, Java a fost și încă este un fenomen neprecedat și extrem de valorificat atât de comunitatea academică cât și de cea comercială. În acest proces, programarea orientată pe obiect s-a dezvoltat ca o paradigmă centrală în industria informatică.

Cum funcționează:

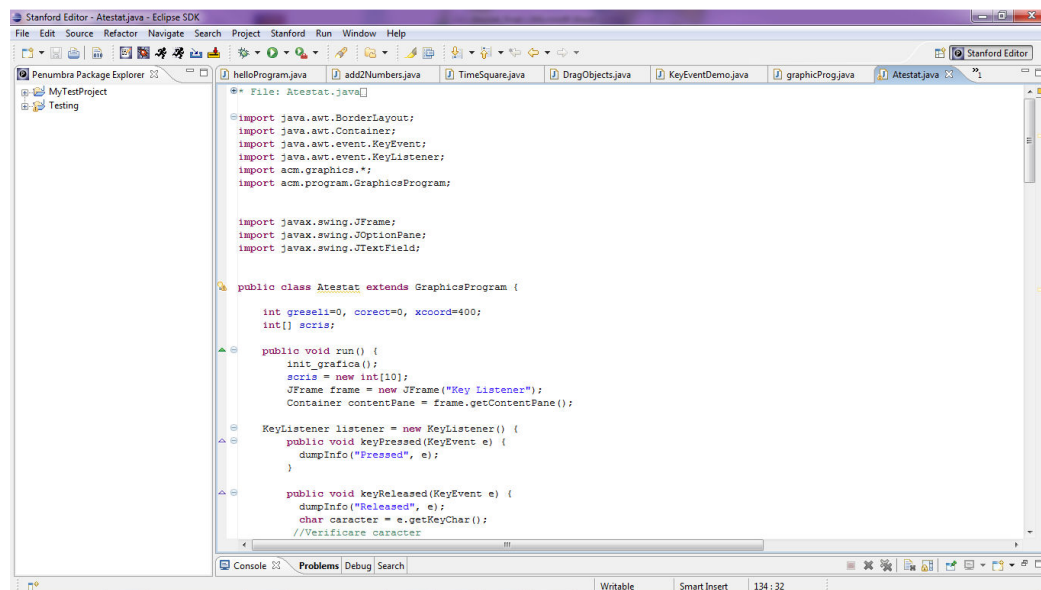


Convenții de notație și Definiții:

Fișier sursă Java (extensie *.java*): este fișierul în care se va afla codul sursă, spre asemenea limbajului de programare C++, extensie a cărui cod sursă este *.cpp/.c*, extensia codului sursă a unui program Java este *.java*

Cod sursă Java (convenție): codul sursă va fi compus din antet (partea de sus a programului în care se vor face legăturile cu fișierele necesare rulării anumitor funcții predefinite Java) spre exemplu, cum C++ folosește procedura *include* pentru a face legătura între program și un header specific anumitei librării care conține funcții necesare în rularea optimă a programului, Java folosește procedura *import*, însă spre deosebire de C++, Java nu suportă operația de linkare între fișiere Java prin *import* (cum C++ face prin *include*, e.g. *#include <fisierheaderseparat.h>*), linkarea având loc prin intermediul procedurii de extindere (*extends*) care va fi decisivă în realizarea unei legături între unu sau mai multe programe Java.

Fișier clasă (convenție): fișierul clasă este un cod sursă Java care va conține declarațiile unei clase care, în funcție de alegerea programatorului, poate fi linkată cu una sau mai multe fișiere clasă diferite.



III. Prima aplicație în Java

Folosind compilatorul oferit de Eclipse SDK vom dezvolta o aplicație care va afișa pe ecran următoarele cuvinte: *Hello World!*

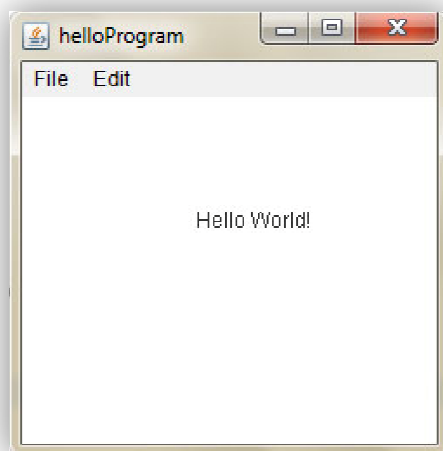
În C++, procedura are loc în felul următor:

```
1  #include <iostream.h>
2
3  int main(void) {
4      cout<<"Hello World";
5  }
```

Iar în Java avem următorul cod:

```
1 import acm.graphics.*;
2 import acm.program.*;
3
4 public class helloProgram extends GraphicsProgram {
5     public void run() {
6         add( new GLabel( "Hello World!", 100, 75 ));
7     }
8 }
```

Iar rezultatul va fi următorul:



Explicații:

Funcția *GLabel* (*Graphical Label*), folosită prin intermediul librăriei *acm.graphics.** are următoarea sintaxă:

GLabel ("Text aici", x, y);

Astfel textul "Text aici" va fi afișat la o distanță x de partea stângă și la o distanță y de partea de sus a ferestrei. Se poate asemana cu coordonatele de tip Height – Width.

Funcția *add()*; are rolul de a afișa pe ecran sau, mai exact, de a adăuga un obiect deja definit – este echivalentul procedurii "*cout*" sau "*printf*" din C++.

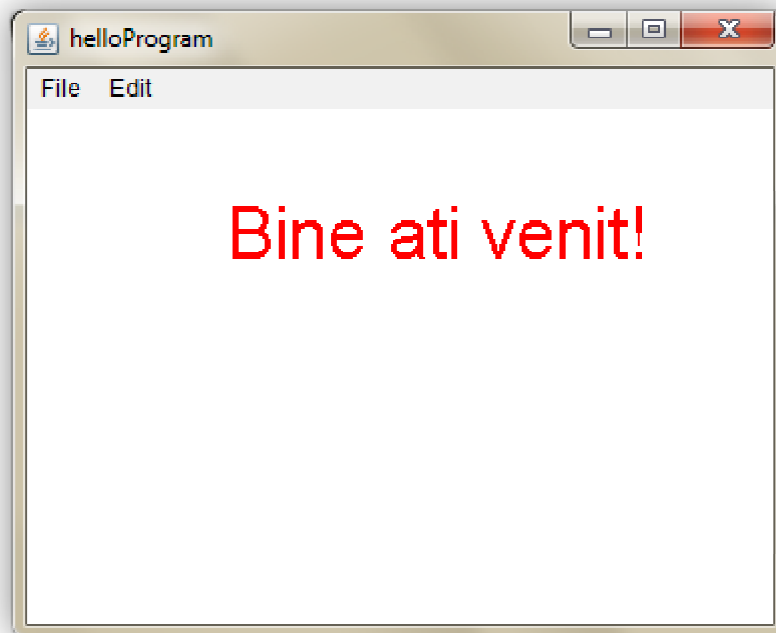
Prin intermediul diverselor funcții programatorul are acces la nenumărate ajustări ale programului său în funcție de necesități.

Să zicem, spre exemplu, că un programator vrea să facă un program simplu care va afișa un text de o anumite mărime, cu un anumit font, și chiar cu o anumită culoare pentru a-l folosi ca text de deschidere a unei aplicații mai complexe.

Programatorul poate ajusta codul de mai surs necesității sale în felul următor:

```
1 import acm.graphics.*;
2 import acm.program.*;
3 import java.awt.*;
4
5 public class helloProgram extends GraphicsProgram {
6     public void run() {
7         GLabel label = new GLabel ("Bine ati venit!", 100, 75);
8         label.setFont("SansSerif-36");
9         label.setColor(Color.RED);
10        add(label);
11    }
12 }
```

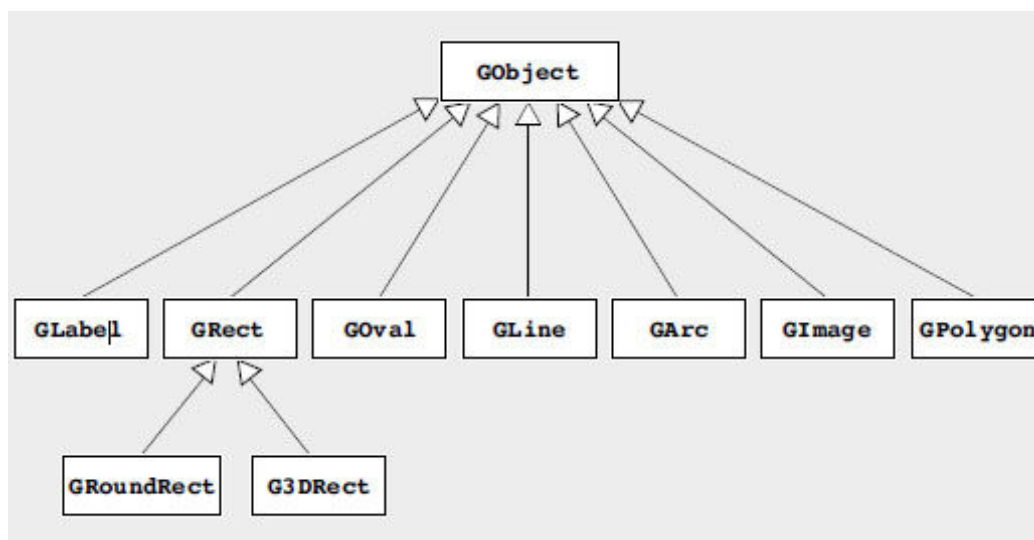
Care va afișa următoarea fereastră:



Se remarcă astfel apariția unui obiect numit *label*, obiectele sunt asemănătoare variabilelor din C++, mai exact structurilor din C++, acestea fiind definite mai exact prin intermediul diverselor proceduri precum *setFont* sau *setColor* care au rolul de a defini toate caracteristicile obiectului specific *GLabel*.

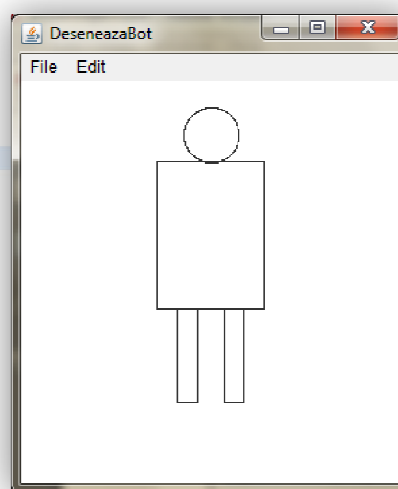
IV. Folosirea instrumentelor grafice în Java

Prin intermediul librăriei *acm.graphics.** avem acces la o arie destul de largă de obiecte cu formă geometrică ce pot fi folosite într-un program. Acestea sunt, după cum urmează:



Un obiect *GRect* desenează un patrulater pe o porțiune din fereastra iar un obiect *GOval* desenează un cerc care se încadrează într-un patrulater după dimensiunile specificate.

```
1 import acm.graphics.*;
2 import acm.program.*;
3
4 public class DeseneazaBot extends GraphicsProgram {
5     public void run() {
6         add(new GOval(120, 20, 40, 40));
7         add(new GRect(100, 60, 80, 110));
8         add(new GRect(115, 170, 15, 70));
9         add(new GRect(150, 170, 15, 70));
10    }
11 }
```



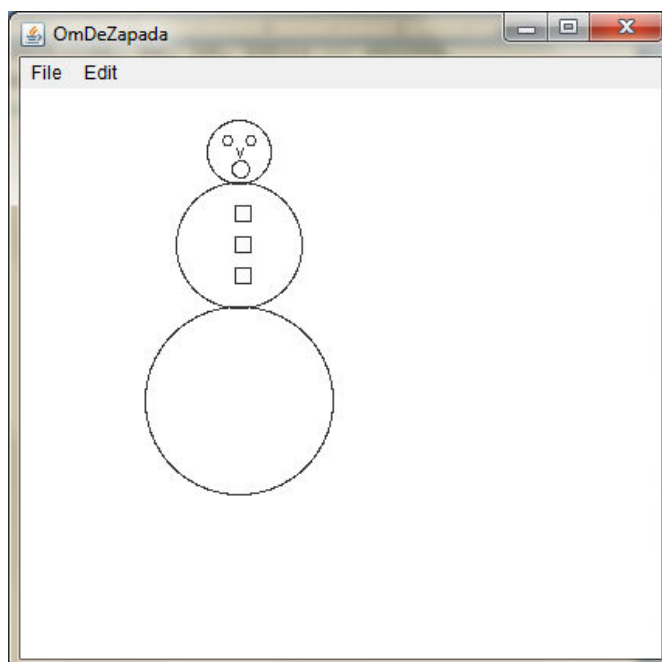
Pentru a concepe o idee mai bună a cum se lucrează cu obiectele vizuale prezente în librăria Java, vom realiza un program, folosindu-ne creativitatea, în care vom desena un om de zăpadă, pornind de la exemplul de mai sus a afișării unui robot.

Soluție: Calculăm coordonatele fiecărui obiect, construim o listă cu obiectele care vor fi necesare și le poziționăm în mod corect. Se remarcă un mic algoritm recursiv prin care anumite obiecte vor fi poziționate fie pe aceleași coordonate x, fie y.

Astfel avem:

```
1 import acm.graphics.*;
2 import acm.program.*;
3
4 public class OmDeZapada extends GraphicsProgram {
5     public void run() {
6         add(new GOval(120, 20, 40, 40));
7         add(new GOval(100, 60, 80, 80));
8         add(new GOval(80, 140, 120, 120));
9
10        // Starts here - Draw Snowman
11        add(new GOval(130, 30, 5, 5)); // Left eye
12        add(new GOval(145, 30, 5, 5)); // Right eye
13        add(new GLabel("v", 139, 45)); // Nose
14        add(new GOval(136, 46, 10, 10)); // Mouth
15        //*****//
16        add(new GRect(138, 75, 10, 10)); // first
17        add(new GRect(138, 95, 10, 10)); // second
18        add(new GRect(138, 115, 10, 10)); // third
19        // ends here
20
21    }
22 }
```

Rezultatul va fi:

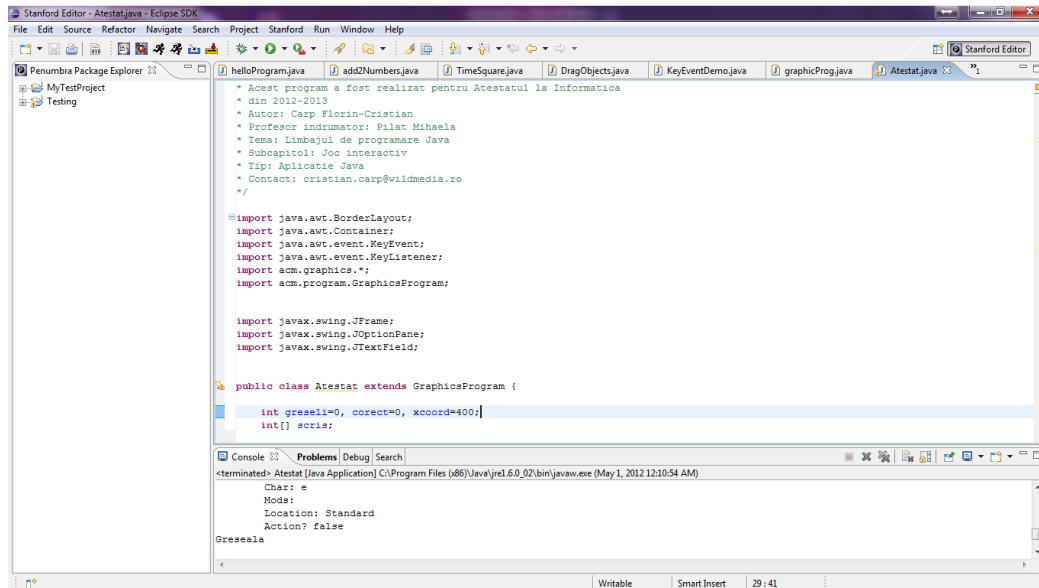


Se remarcă folosirea atât a obiectelor grafice de tip *GRect*, *GOval*, cât și cele de tip Text: *GLabel*.

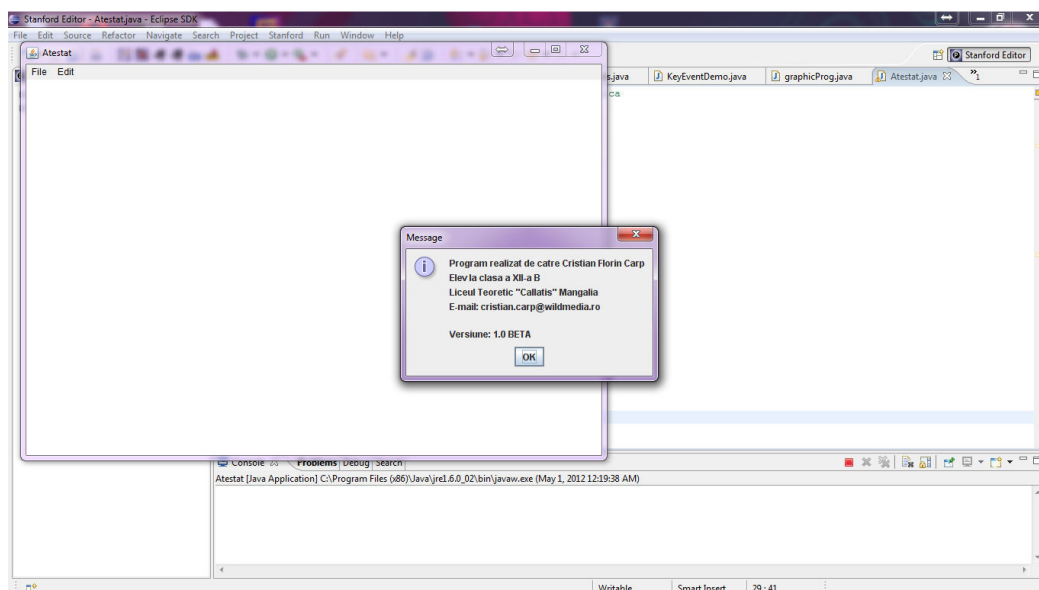
Astfel, cu un compiler Java și cu librăria *acm.jar* inclusă oricine poate aduce modificări la acest om de zăpadă, aceștia având posibilitatea să-l și animeze.

Capitolul V: Joc interactiv

Am ales să-mi testez cunoștințele Java prin crearea unui joc interactiv de tip HangMan (spânzurătoarea), astfel, cu ajutorul instrumentelor grafice și a unui număr minimal de variabile am reușit să implementez un joc cu grafică vizuală și care interacționează cu utilizatorul de tip spânzurătoarea.



Am creat acest joc prin intermediul editorului Eclipse SDK, cu ajutorul librăriei adiționale acm.jar care mi-a permis afișarea grafică a tuturor elementelor necesare.



Structură aplicație

Aplicația se structurează într-un program complex realizat în Java, declarat sub numele *Atestat*

```
public class Atestat extends GraphicsProgram {
```

Programul principal este structurat în următoarele funcții și proceduri:

init_grafica(); - această funcție este apelată la începutul rulării programului, cu rolul de a desena pe ecran grafica propriu-zisă.

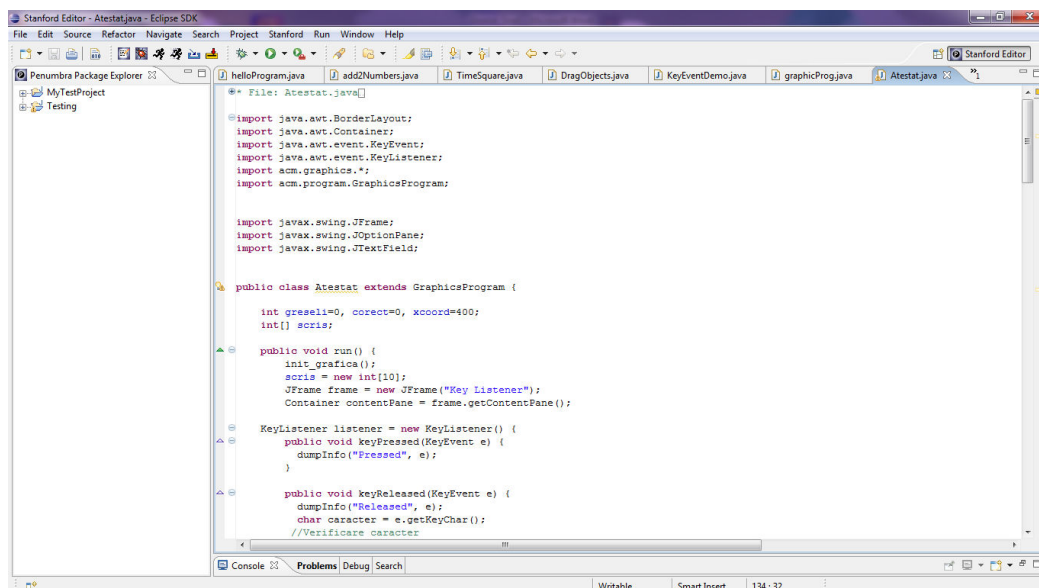
KeyListener listener, keyPressed(), keyReleased(), keyTyped() – aceste funcții au rolul de a observa interacțiunea utilizatorului cu tastatura, astfel, se remarcă dacă utilizatorul apasă o tastă – care este aceea – dacă convine sau nu condiției.

verific(char character); - cu rolul de a verifica dacă caracterul trimis este valid sau nu – dacă se încadrează în condiția de existență sau nu

greseala(int numar); - cu rolul de a-l penaliza pe utilizator când greșește, sau, în caz de final, când termină jocul fără să câștige.

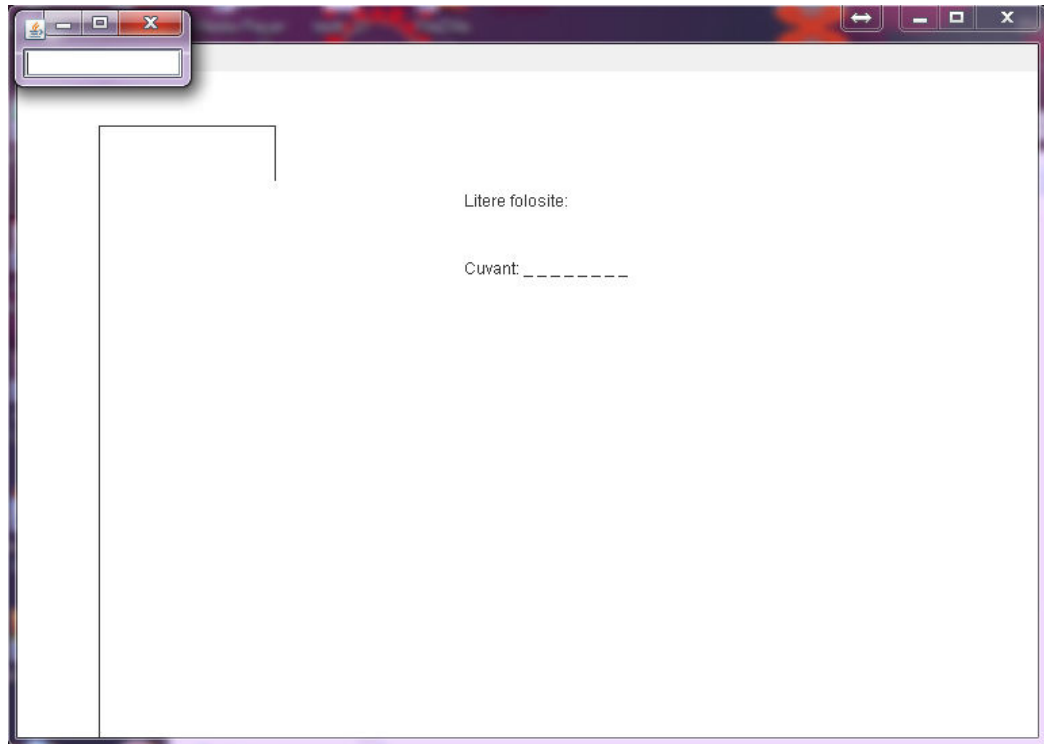
alert(String string); - cu rolul de a afișa un alert box pentru utilizator

Prin intermediul alert box-urilor oferite din structura standard a Java am reușit să implementez o notificare permanentă cu utilizatorul/player-ul efectiv.



Detalii tehnice de implementare

Interfața grafică a fost creată prin intermediul desenării elementelor geometrice pe ecran, astfel, am utilizat elementele de tip *GLine* pentru afișarea tuturor liniilor pentru interfața de startup (pregătirea “spânzurătoarei”), iar prin intermediul *GLabel* am afișat căsuțele în care cuvintele/literele din joc vor fi afișate, atât literele corecte cât și cele incorecte.



Prin intermediul procedurii de tip *Event Listener* am implementat un *dialog box* cu rolul de a capta orice *event* venit din partea tastaturii, practic, acest *Event Listener* are rol de observator în ceea ce privește acțiunile efectuate de utilizator/player la nivelul tastaturii.

```
public void run() {
    init_grafica();
    scris = new int[10];
    JFrame frame = new JFrame("Key Listener");
    Container contentPane = frame.getContentPane();

    KeyListener listener = new KeyListener() {
        public void keyPressed(KeyEvent e) {
            dumpInfo("Pressed", e);
        }
    }
```

Prin intermediul funcției *verific()* am implementat un sistem de verificare a literelor/caracterelor trimise de utilizator, astfel, acestea pot fi corecte, aflându-se în tiparul de rezolvare, sau incorecte, utilizatorul fiind penalizat ca atare.

```
public void verific(char caracter) {

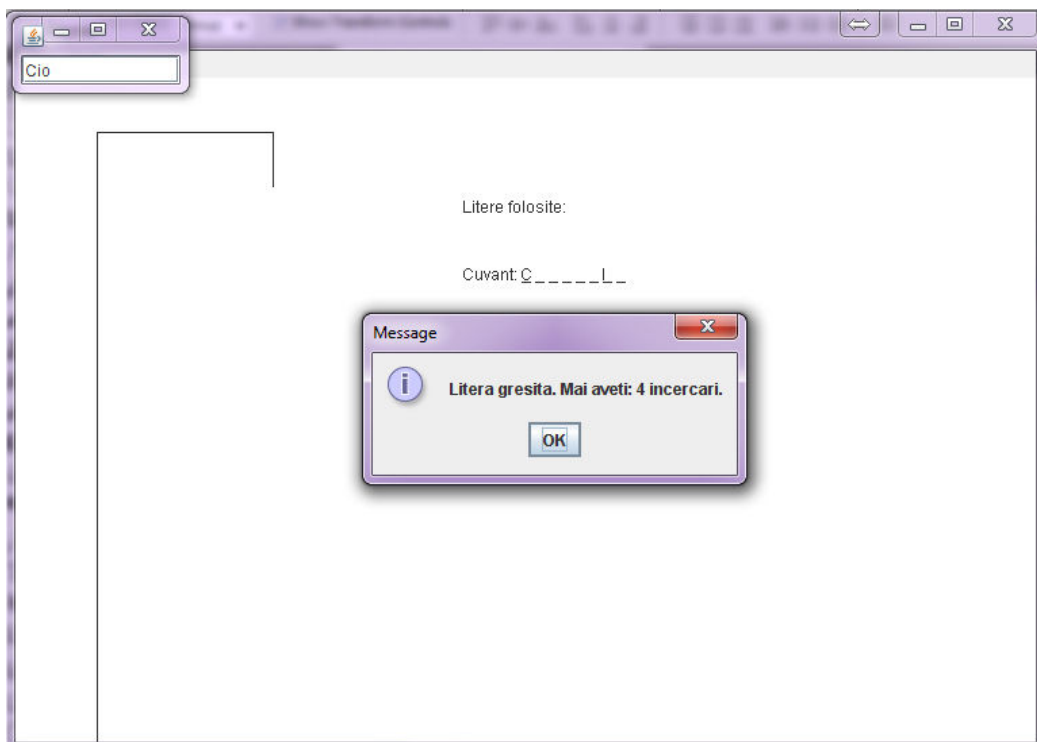
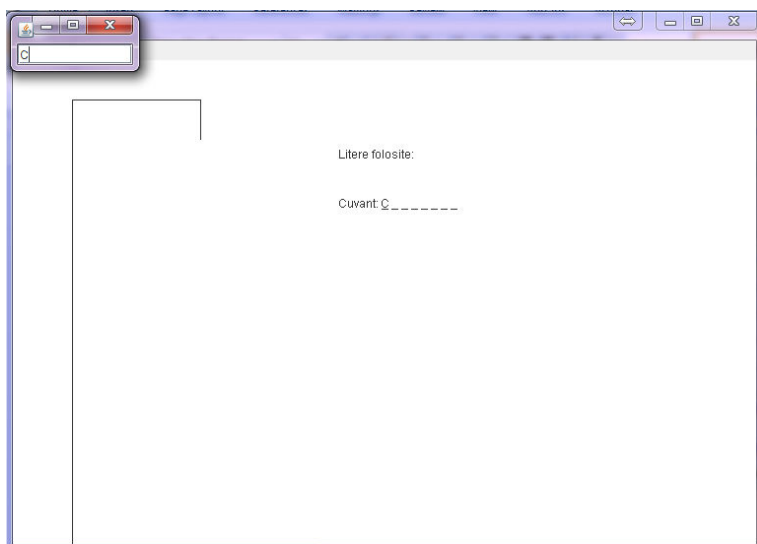
    switch (caracter) {
        case 'a': case 'A':
            add(new JLabel("A", 383, 150));
            add(new JLabel("A", 413, 150));
            println(corect);
            if (++scris[1] == 1)
                corect+=2;
            break;
        case 'l': case 'L':
            add(new JLabel("L", 393, 150));
            add(new JLabel("L", 403, 150));
            if (++scris[2] == 1)
                corect+=2;
            break;
        case 'c': case 'C':
            add(new JLabel("C", 373, 150));
            if (++scris[3] == 1)
                ++corect;
            break;
        case 't': case 'T':
            add(new JLabel("T", 423, 150));
            if (++scris[4] == 1)
                ++corect;
            break;
        case 'i': case 'I':
            add(new JLabel("I", 433, 150));
            if (++scris[5] == 1)
                ++corect;
            break;
        case 's': case 'S':
            add(new JLabel("S", 443, 150));
            if (++scris[6] == 1)
                ++corect;
            break;
        case 'b': case 'd': case 'e': case 'f': case 'g':
        case 'h':
        case 'j': case 'k': case 'm': case 'n': case 'p':
        case 'q':
        case 'r': case 'u': case 'v': case 'w': case 'o':
        case 'x':
        case 'y': case 'z': case 'B': case 'D': case 'E':
        case 'F':
        case 'G': case 'H': case 'J': case 'K': case 'M':
        case 'N':
        case 'P': case 'Q': case 'R': case 'U': case 'V':
        case 'W':
        case 'O': case 'X': case 'Y': case 'Z':
            String mesaj = "Litera gresita. Mai aveti: " + (4-
greseli) + " incercari.";
            alert(mesaj);
            System.out.println("Greseala");
            xcoord+=10;
            add(new JLabel("" + caracter, xcoord, 100));
    }
```

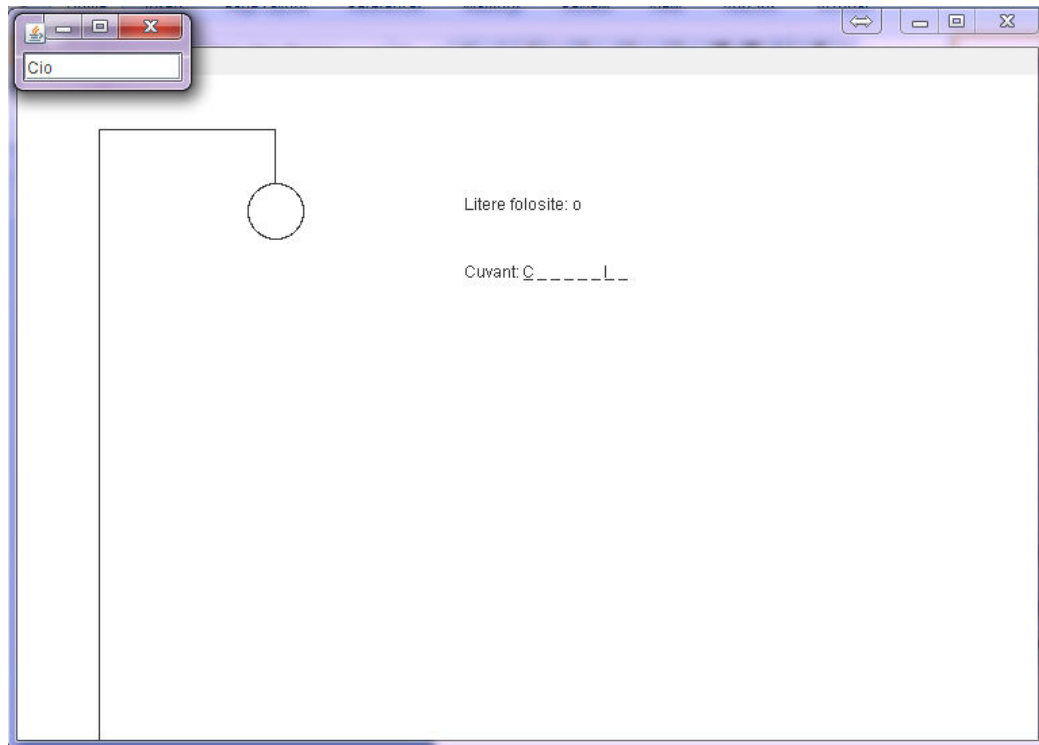
```

        greseala(++greseli);
        break;
    }
    if (corect >= 8) {
        alert("Felicitari, ati castigat!");
        exit();
    }
}

```

În exemplul de față, cuvântul corect este *CALLATIS*, simulând un joc real, utilizatorul tastează întâi litera C, apoi I, pentru ca mai apoi sa tasteze O (ceea ce este incorect), programul va acționa astfel:

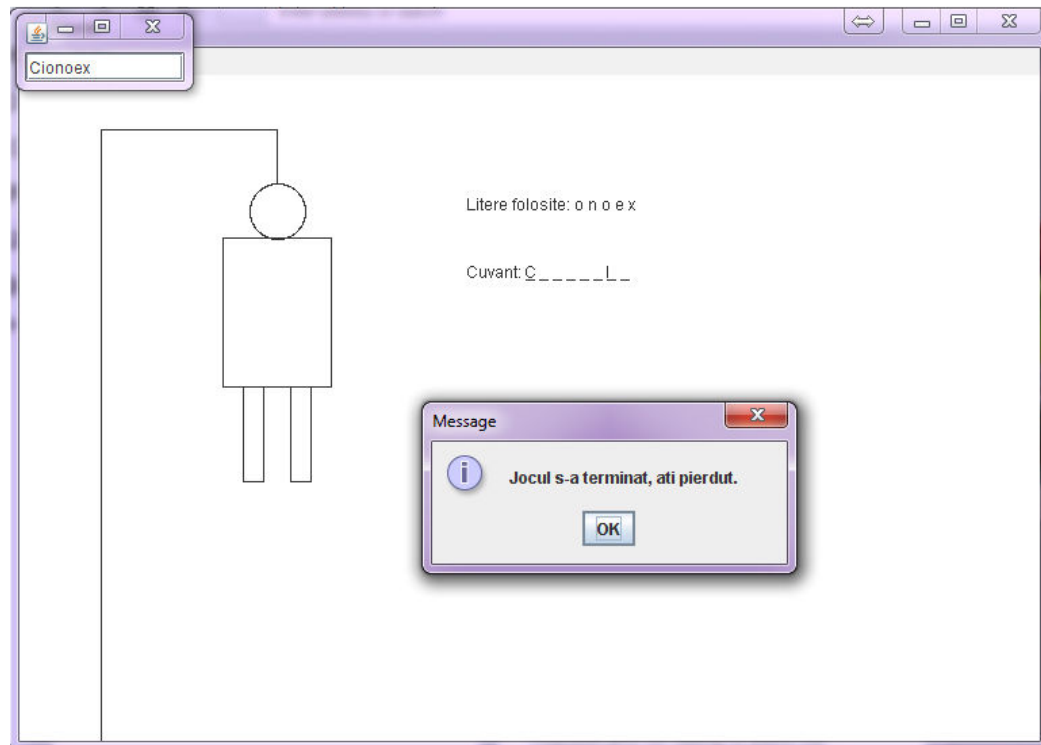




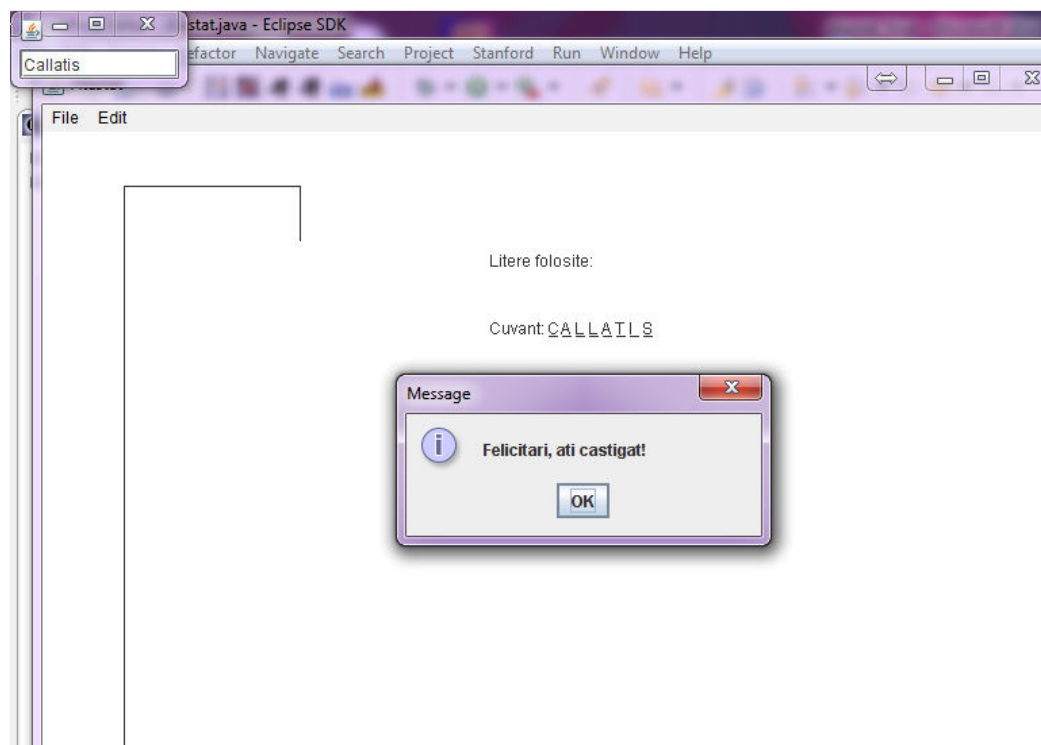
În acest moment se adaugă primul element din spânzurătoarea propriuzisă. Jucătorul are dreptul la maxim 5 greșeli.

Dacă jucătorul va epuiza toate încercările (în urma apelării funcției *greseala()*), acesta va fi scos din joc automat.

```
public void greseala(int numar) {  
    // deseneaza cand greseste  
    switch (numar) {  
        case 1:  
            add(new GOval(170, 80, 40, 40)); //cap  
            break;  
        case 2:  
            add(new GRect(150, 120, 80, 110)); //corp  
            break;  
        case 3:  
            add(new GRect(165, 230, 15, 70)); //picior 1  
            break;  
        case 4:  
            add(new GRect(200, 230, 15, 70)); //picior 2  
            break;  
        case 5:  
            alert("Jocul s-a terminat, ati pierdut.");  
            println("Game over.");  
            exit();  
            break;  
    }  
}
```



În cazul în care acesta va reuși să completeze toate literele corespunzător jocul va lua sfârșit iar utilizatorul/player-ul va fi felicitat pentru câștig.



Resurse hardware și software

Pentru buna funcționare a jocului interactiv sunt prefigurate următoarele cerințe minime:

Sistem de operare: Windows XP/Vista/7/2003 Server; Linux

Programe suplimentare: Java Runtime,
Java SDK

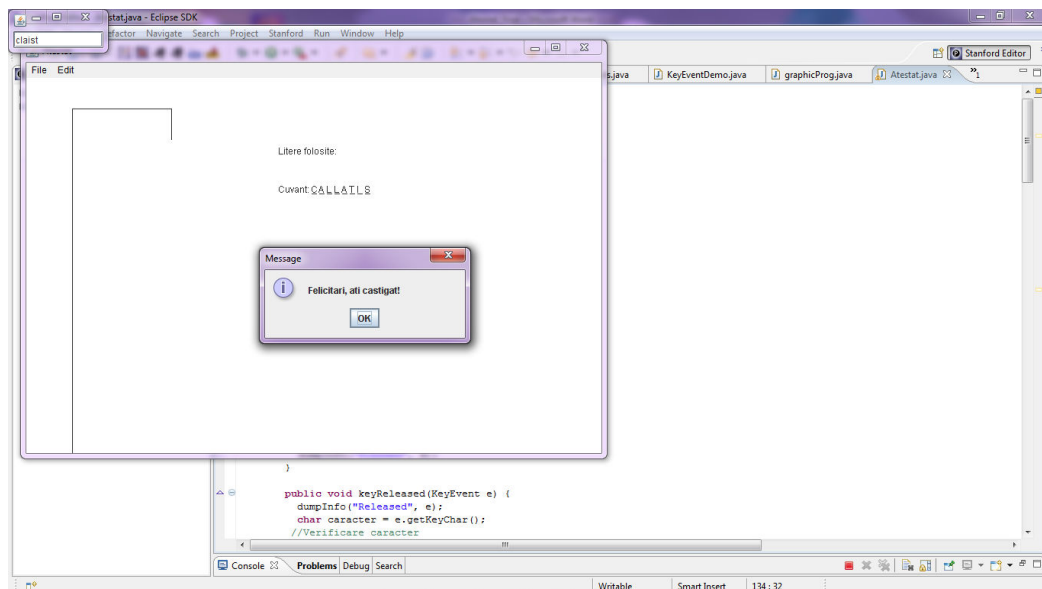
Cerințe hard minime: 128 MB RAM,

Placă video integrată

Cerințe hard recomandate: 1 GB RAM,

Placă video integrată

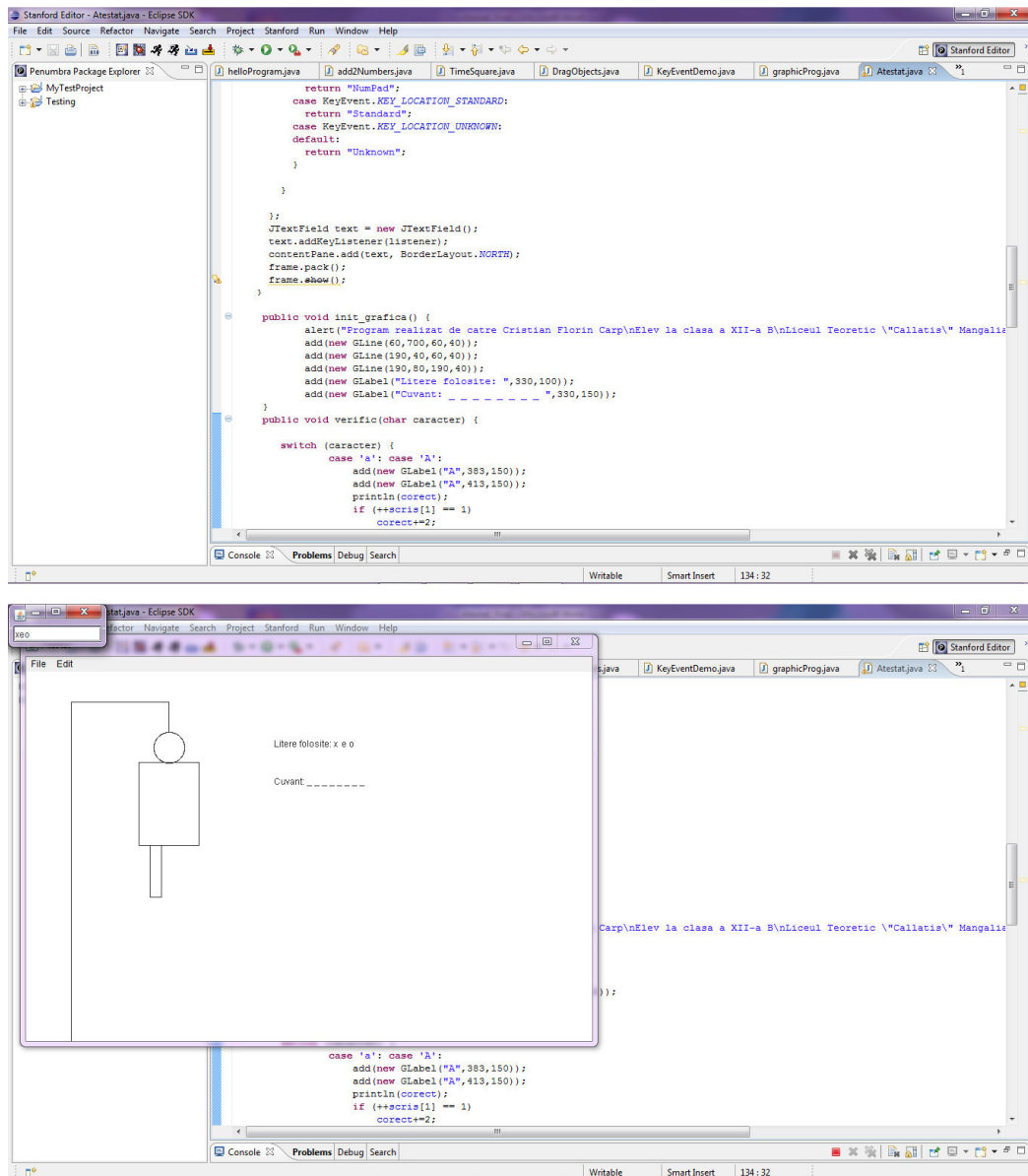
Deoarece aplicația nu necesită alocarea de memorie foarte mare se remarcă fiabilitatea programului în materie de funcționarea sistemului de operare a multi-taskingului, jocul interactiv consumând minimul necesar pentru buna funcționare a acestuia.



Modalități de utilizare

Aplicația este pornită ca orice altă aplicație. Utilizatorul/player-ul trebuie să tasteze literele/caracterele propriuzise în căsuța de dialog. Utilizatorul va fi întotdeauna notificat dacă greșește de asemenea și în cazul în care acesta câștigă.

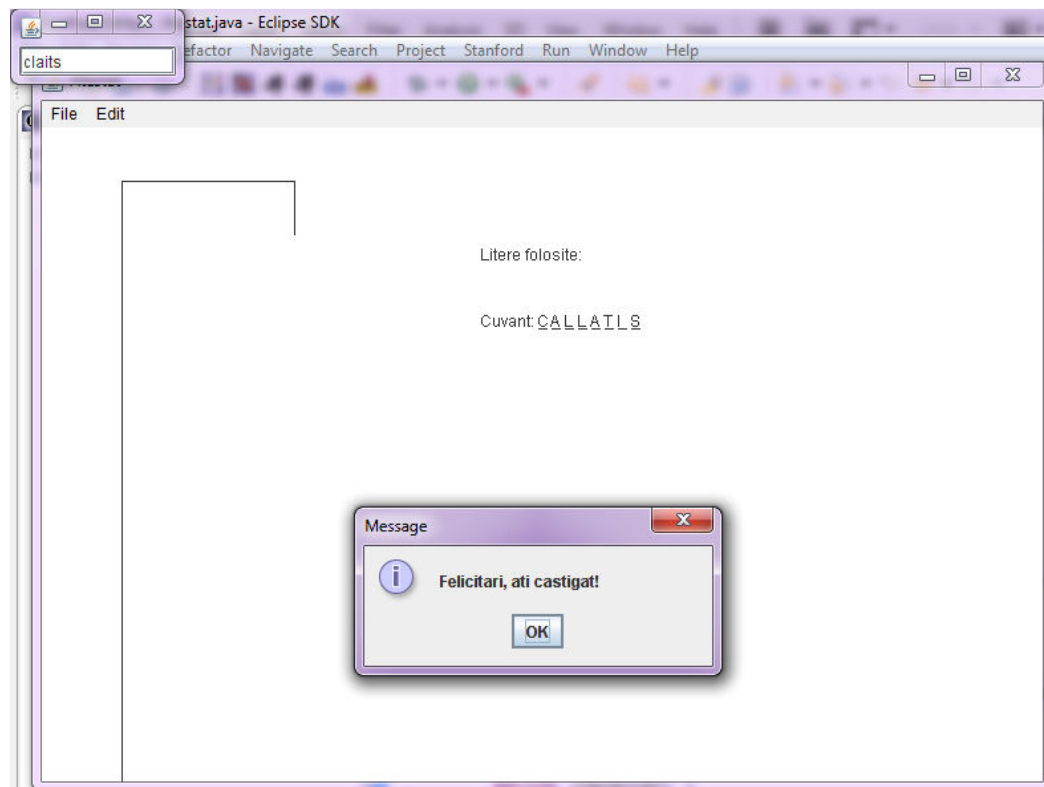
Această aplicație poate fi folosită atât în mediul local cât și pe internet prin intermediul implementării într-o pagină web.



Modalități de dezvoltare

Programul poate fi dezvoltat în diferite moduri, se pot implementa funcții suplimentare cu rolul de a randomiza alegerea unor cuvinte pentru joc dintr-o bază de date, implementată la rândul ei prin intermediul unui fișier secundar care, prin intermediul unei funcții de citire va furniza alte cuvinte.

Interfața utilizatorului poate fi schimbată în orice moment prin implementarea de animații interactive cu utilizatorul sau chiar a unei grafice 3D.



Cod Sursă

```
/*
 * File: Atestat.java
 * -----
 * Acest program a fost realizat pentru Atestatul la Informatica
 * din 2012-2013
 * Autor: Carp Florin-Cristian
 * Profesor indrumator: Pilat Mihaela
 * Tema: Limbajul de programare Java
 * Subcapitol: Joc interactiv
 * Tip: Aplicatie Java
 * Contact: cristian.carp@wildmedia.ro
 */

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import acm.graphics.*;
import acm.program.GraphicsProgram;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class Atestat extends GraphicsProgram {

    int greseli=0, corect=0, xcoord=400;
    int[] scris;

    public void run() {
        init_grafica();
        scris = new int[10];
        JFrame frame = new JFrame("Key Listener");
        Container contentPane = frame.getContentPane();

        KeyListener listener = new KeyListener() {
            public void keyPressed(KeyEvent e) {
                dumpInfo("Pressed", e);
            }

            public void keyReleased(KeyEvent e) {
                dumpInfo("Released", e);
                char caracter = e.getKeyChar();
                //Verificare caracter
                verific(caracter);
            }

            public void keyTyped(KeyEvent e) {
                dumpInfo("Typed", e);
            }

            private void dumpInfo(String s, KeyEvent e) {
                System.out.println(s);
                int code = e.getKeyCode();
                System.out.println("\tCode: " + KeyEvent.getKeyText(code));
            }
        };
    }
}
```

```

        System.out.println("\tChar: " + e.getKeyChar());
        int mods = e.getModifiersEx();
        System.out.println("\tMods: "
            + KeyEvent.getModifiersExText(mods));
        System.out.println("\tLocation: "
            + location(e.getKeyLocation()));
        System.out.println("\tAction? " + e.isActionKey());
    }

    private String location(int location) {
        switch (location) {
            case KeyEvent.KEY_LOCATION_LEFT:
                return "Left";
            case KeyEvent.KEY_LOCATION_RIGHT:
                return "Right";
            case KeyEvent.KEY_LOCATION_NUMPAD:
                return "NumPad";
            case KeyEvent.KEY_LOCATION_STANDARD:
                return "Standard";
            case KeyEvent.KEY_LOCATION_UNKNOWN:
            default:
                return "Unknown";
        }
    }

};
JTextField text = new JTextField();
text.addKeyListener(listener);
contentPane.add(text, BorderLayout.NORTH);
frame.pack();
frame.show();
}

    public void init_grafica() {
        alert("Program realizat de catre Cristian Florin
Carp\nElev la clasa a XII-a B\nLiceul Teoretic \"Callatis\"
Mangalia\nE-mail: cristian.carp@wildmedia.ro\n\nVersiune: 1.0 BETA");
        add(new GLine(60,700,60,40));
        add(new GLine(190,40,60,40));
        add(new GLine(190,80,190,40));
        add(new GLabel("Litere folosite: ",330,100));
        add(new GLabel("Cuvant: _ _ _ _ _ ",330,150));
    }

    public void verific(char caracter) {

        switch (caracter) {
            case 'a': case 'A':
                add(new GLabel("A",383,150));
                add(new GLabel("A",413,150));
                println(corect);
                if (++scris[1] == 1)
                    corect+=2;
                break;
            case 'l': case 'L':
                add(new GLabel("L",393,150));
                add(new GLabel("L",403,150));
                if (++scris[2] == 1)
                    corect+=2;
                break;
            case 'c': case 'C':

```

```

        add(new GLabel("C", 373, 150));
        if (++scris[3] == 1)
            ++corect;
        break;
    case 't': case 'T':
        add(new GLabel("T", 423, 150));
        if (++scris[4] == 1)
            ++corect;
        break;
    case 'i': case 'I':
        add(new GLabel("I", 433, 150));
        if (++scris[5] == 1)
            ++corect;
        break;
    case 's': case 'S':
        add(new GLabel("S", 443, 150));
        if (++scris[6] == 1)
            ++corect;
        break;

    case 'b': case 'd': case 'e': case 'f': case 'g':
    case 'h':
    case 'j': case 'k': case 'm': case 'n': case 'p':
    case 'q':
    case 'r': case 'u': case 'v': case 'w': case 'o':
    case 'x':
    case 'y': case 'z': case 'B': case 'D': case 'E':
    case 'F':
    case 'G': case 'H': case 'J': case 'K': case 'M':
    case 'N':
    case 'P': case 'Q': case 'R': case 'U': case 'V':
    case 'W':
        case 'O': case 'X': case 'Y': case 'Z':
        String mesaj = "Litera gresita. Mai aveti: " + (4-
greseli) + " incercari.";
        alert(mesaj);
        System.out.println("Greseala");
        xcoord+=10;
        add(new GLabel("" + caracter, xcoord, 100));
        greseala(++greseli);
        break;
    }
    if (corect >= 8) {
        alert("Felicitari, ati castigat!");
        exit();
    }
}

}

public void greseala(int numar) {
    // deseneaza cand greseste
    switch (numar) {
        case 1:
            add(new GOval(170, 80, 40, 40)); //cap
            break;
        case 2:
            add(new GRect(150, 120, 80, 110)); //corp
            break;
        case 3:
            add(new GRect(165, 230, 15, 70)); //picior 1
            break;
    }
}

```

```

        case 4:
            add(new GRect(200, 230, 15, 70)); //picior 2
            break;
        case 5:
            alert("Jocul s-a terminat, ati pierdut.");
            println("Game over.");
            exit();
            break;
    }
}

public void alert(String string) {
    JFrame alert = null;
    JOptionPane.showMessageDialog(alert, "" + string);
}
}

```

Bibliografie

www.java.com

www.oracle.com

www.stackoverflow.com

www.dreamincode.net

www.stanford.edu

www.stanford.edu/class/cs108/