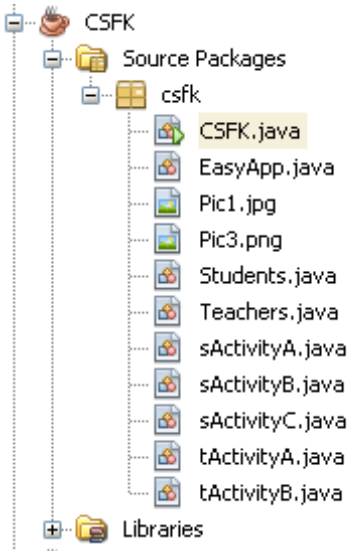
Criterion C: Development

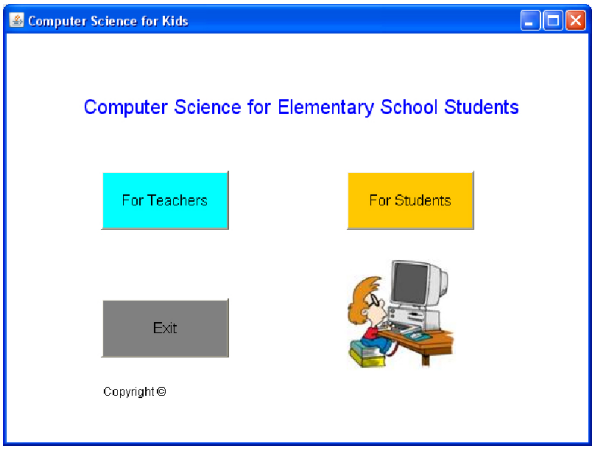
CSFK is Java based program for teacher (client Ms xx) and students; teacher creates activities/tasks with answers; program saves them automatically in .txt file as one record; program creates tasks/test for students; checks answers and gives results for each task; in case of error gives chance to correct it;

CSFK file should be located in folder where ActivityA.txt and ActivityB.txt will be created. Teacher can print created activities any time.



Program's structure

Main window:

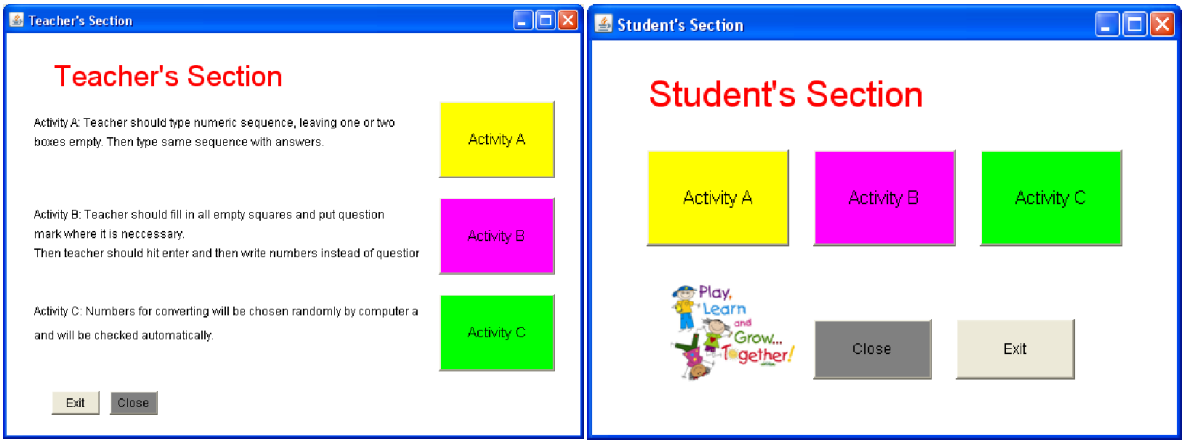


"For Teachers" is option for client to create Activity_A or Activity_B (Activity_C-generated by program); "For Students"- option for student to solve tasks.

I used EasyApp.java class to create different Java AWT elements: Labels, TextFields, Buttons, Lists for interface; ImageIcon and JLabel components to put pictures in window.

```
Label Title = addLabel("Computer Science for Elementary School Students", 80, 80, 500, 50, this);
Button Teachers = addButton("For Teachers", 100, 170, 130, 60, this);
Button Students = addButton("For Students", 350, 170, 130, 60, this);
Button Exit = addButton("Exit", 100, 300, 130, 60, this);
Label CopyRight = addLabel("Copyright © Tbilisi 2013", 100, 370, 200, 50, this);
ImageIcon Icon1 = new ImageIcon(getClass().getResource("Pic1.jpg"));
JLabel Pic1 = addJLabel(Icon1, 350, 260, 111, 115, this);

public CSFK() {
    setTitle("Computer Science for Kids");
    Title.setForeground(Color.blue);
    Title.setFont(new Font("Arial", 0, 20));
    Teachers.setFont(new Font("Arial", 0, 15));
    Students.setFont(new Font("Arial", 0, 15));
    Exit.setFont(new Font("Arial", 0, 15));
    setBounds(50, 50, 600, 450);
    Teachers.setBackground(Color.CYAN);
    Students.setBackground(Color.orange);
    Exit.setBackground(Color.GRAY);
}
```

All classes extend EasyApp (Mulkey, 2004-2007)[1];

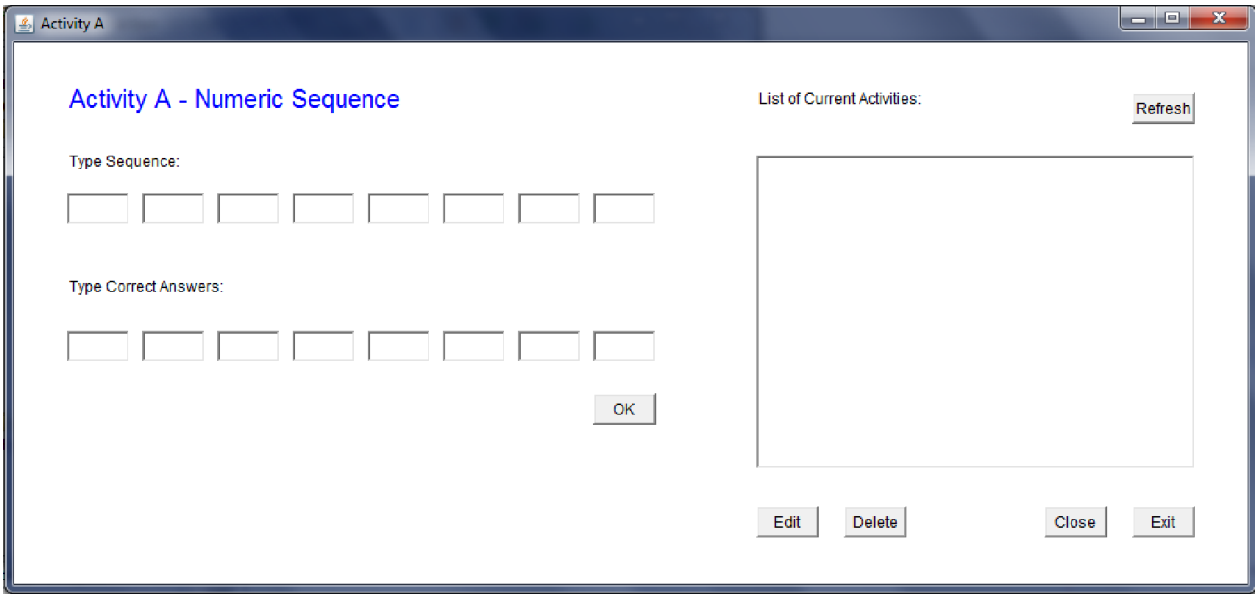


Teacher's and student's section window after clicking buttons in main window.

I created activity description for teacher.

[1]Mulkey, D. (2004-2007). *Free and Easy Java.* Retrieved xx, from http://ibcomp.fis.edu/easyJava/

Activity A – Teacher's Section – tActivityA.java



tActivityA() class constructor works with RandomAccessFile; if file doesn't exist, it will be created with title in it.

```
public tActivityA() {
    setTitle("Activity A");
    Title.setForeground(Color.blue);
    Title.setFont(new Font("Arial", 0, 20));
    setBounds(50, 50, 1000, 470);

    try {
        RandomAccessFile tActivityA = new RandomAccessFile("ActivityA.txt", "rw");
        if (tActivityA.length() == 0) {
            tActivityA.writeBytes("List of current sequences:\n");
            // If File is empty Java will write List Title on the first line
        }
    } catch (IOException e) {
        e.getMessage();
    }
}
```

Two sequences are typed in textfields:

- **Type Sequence**-3,6,9,12,?,18,?,24
- **Type Correct Answers**-3,6,9,12,15,18,21,24

After pressing button "**OK**", first 8 boxes are gathered under one **String problem** (using method **addActivity()**). Text written in second 8 boxes-under one **String answer.**

```
String problem = pr1 + " " + pr2 + " " + pr3 + " " + pr4 + " " + pr5 +
        " " + pr6 + " " + pr7 + " " + pr8;
String answer = an1 + " " + an2 + " " + an3 + " " + an4 + " " + an5 +
        " " + an6 + " " + an7 + " " + an8 + " ";
```

Where

```
String pr1 = T1.getText();
String an1 = C1.getText();
```

In case of empty box, message warns teacher to fill blanks.

```
if (pr1.equals("") || pr2.equals("") || pr3.equals("") || pr4.equals("")
        || pr5.equals("") || pr6.equals("") || pr7.equals("") || pr8.equals("")
        || an1.equals("") || an2.equals("") || an3.equals("") || an4.equals("")
        || an5.equals("") || an6.equals("") || an7.equals("") || an8.equals("")) {
    outputString("Error,\n" + "Please, fill all fields!");
```

Problem-answer will be added in file **ActivityA.txt** on one line;

```
tActivityA.seek(tActivityA.length());
tActivityA.writeBytes(problem + " - " + answer + "\n");
Clean();
tActivityA.close();
```

Method **addActivity,** using method **Clean()** makes boxes blank. Button **Refresh** displayes current activities in list. All previously written activities are removed by method List.removeAll().

```
public void Refresh() {
    List.removeAll();
    createArray();
    boolean flag = true;
    List.add("Problem List: ");
    List.add("");
```

Method createArray creates array of problems and answers by reading from File; each line is divided in 8+8 elements and saved in array ListA[30][16].

```
static String[][] ListA = new String[30][16];//max 30 activities will be created
        // with 16 elements in each rows; 1-8 represents activities with ?,
        // 9-16 acitivities with all numbers=answer
static int k; //k equals the number of rows that are fill in the array

public static void createArray() {
    try {
        RandomAccessFile tActivityA = new RandomAccessFile("ActivityA.txt", "rw");
```

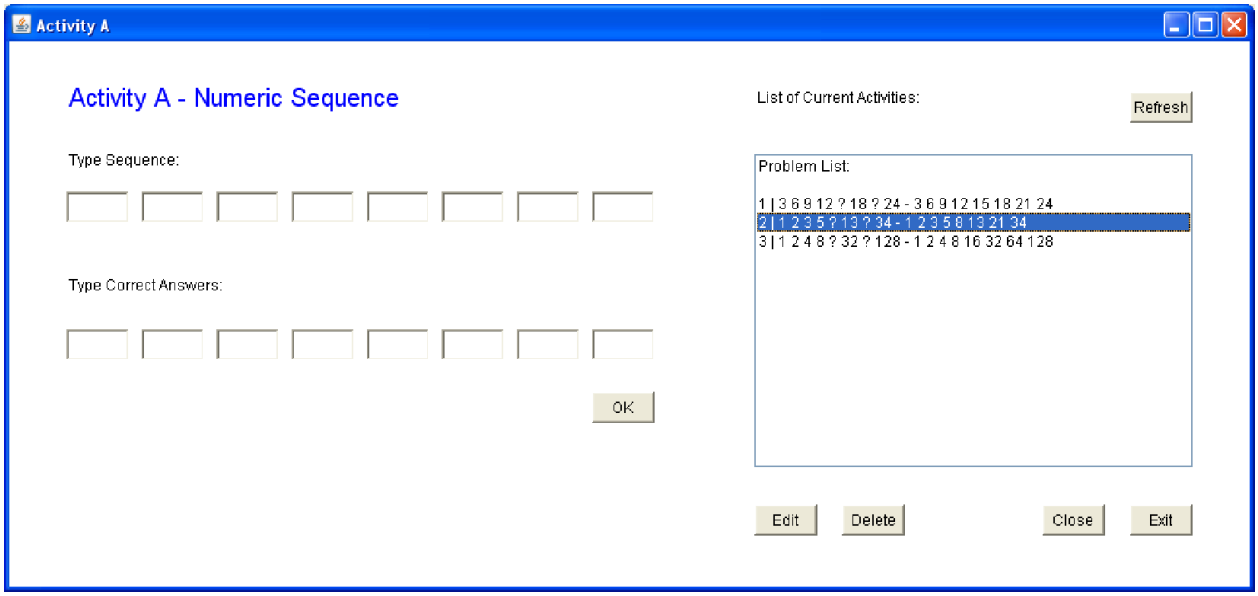
```
156                 String problem = tActivityA.readLine();
157                 if (problem != null) {
158                     for (int i = 0; i < 16; i++) {
159                         if (i != 8) {
160                             x2 = problem.indexOf(" ", x1);
161                         } else {
162                             x1 = x1 + 2;
163                             x2 = problem.indexOf(" ", x1);
164                         }
165
166                         number = problem.substring(x1, x2);
167                         ListA[k][i] = number;
168                         x1 = x2 + 1;
169                     }
```

To display all already created activities in list method **Refresh()** writes first 7 numbers with space; after 8th number it makes "-" to distinguish problems from answers.

```
190                 if (ListA[i][j] != null) {
191                     if (j != 7) {
192                         line = line + ListA[i][j] + " ";
193                     } else {
194                         line = line + ListA[i][j] + " - ";
```

After pressing button **Edit**, method **EditB()** displays selected and deletes item in list.

Edit=Delete+Add;



In List, tasks/activities are numbered to “guess” index of row in array;

```
public void EditB() {
    String line = List.getSelectedItem();
    int x = line.indexOf("|");
    int index = Integer.parseInt(line.substring(0, x - 1)) - 1;
    T1.setText(ListA[index][0]);
    T2.setText(ListA[index][1]);
```

After editing, teacher presses button **OK**; it does same as in adding activity at the begging.

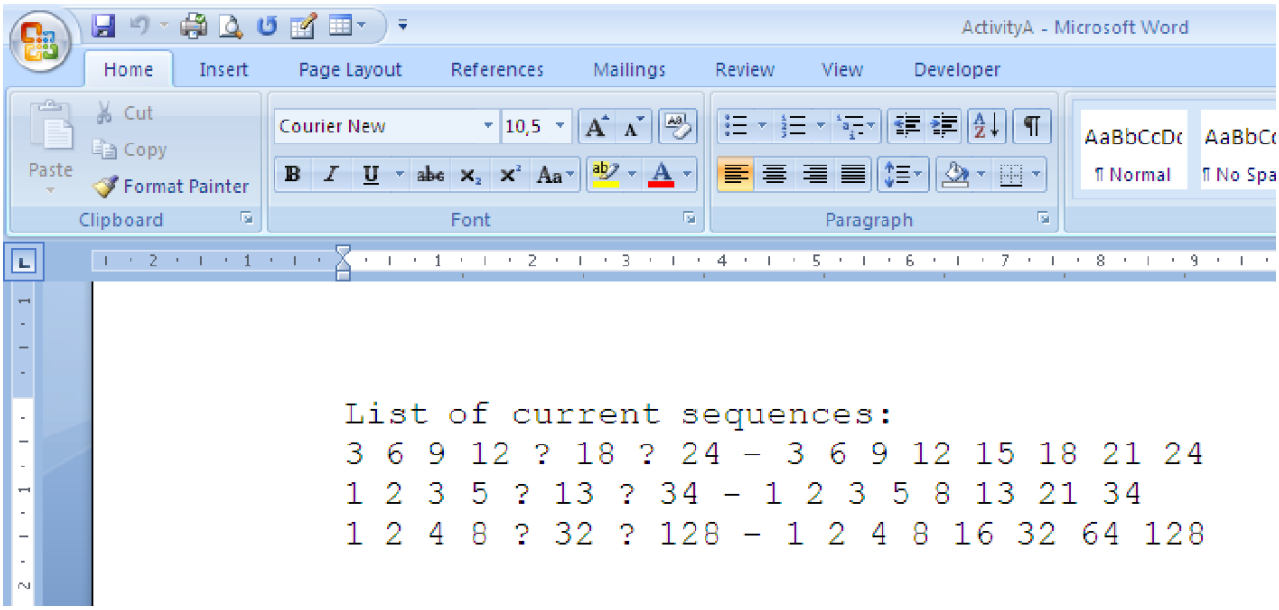
Button **Delete**, deletes activity from array and fills last raw with null.

```
void Delete() {
    String wordDel = List.getSelectedItem();
                        //word/line, we need to delete from array and file
    int x = wordDel.indexOf("|");
    int index = Integer.parseInt(wordDel.substring(0, x - 1)) - 1;
                            //element, we want to delete - index in the array
    int wordlength = wordDel.length();
    //delete word from array
    for (int i = index; i < k - 1; i++) {
        for (int j = 0; j < 16; j++) {
            ListA[i][j] = ListA[i + 1][j];
        }
    }
    //last element in the array fill manualy
    for (int j = 0; j < 16; j++) {
        ListA[k - 1][j] = null;
    }
```

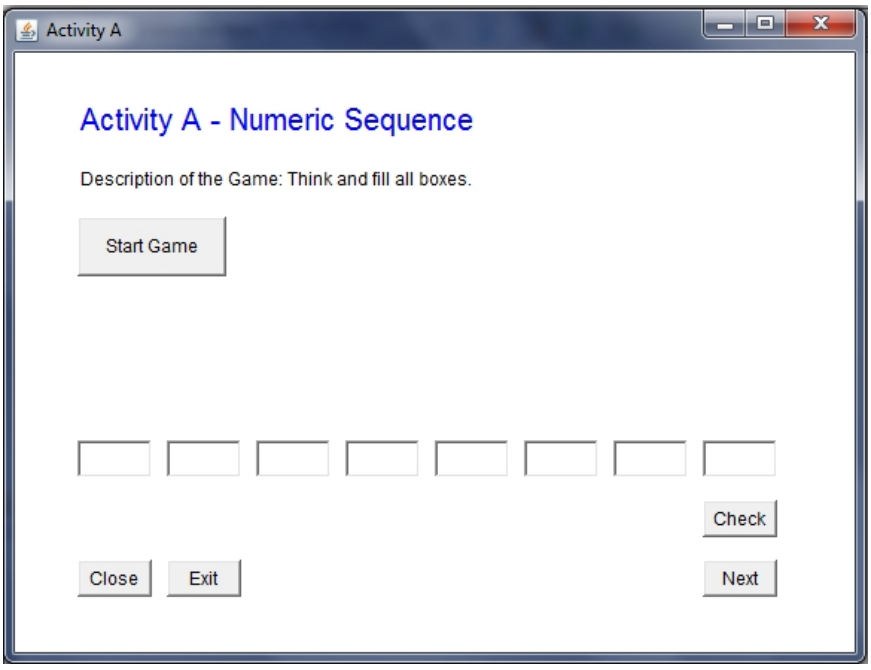
Useless records are cut from array.

```
try {          // change file when a activity is deleted
   RandomAccessFile tActivityA = new RandomAccessFile("ActivityA.txt", "rw");
       long length = tActivityA.length();
                           // file length before deleting a record/activity
       length = length - wordlength;
                             // file length after deleting a record/activity
       tActivityA.readLine();

       for (int p = 0; p < k - 1; p++) {
                    //k is number of records/activities in the file
           String line = "";
           for (int j = 0; j < 16; j++) {
               if (j != 7) {
                   line = line + ListA[p][j] + " ";
                               //reading elements from array
               } else {
                   line = line + ListA[p][j] + " - ";
               }
           }

           tActivityA.writeBytes(line + "\n");
       }
       tActivityA.setLength(length + 2);    // to cut useless records
       tActivityA.writeBytes("\n");
```

Content of ActivityA.txt:



Activity A – student's section – sActivityA.java

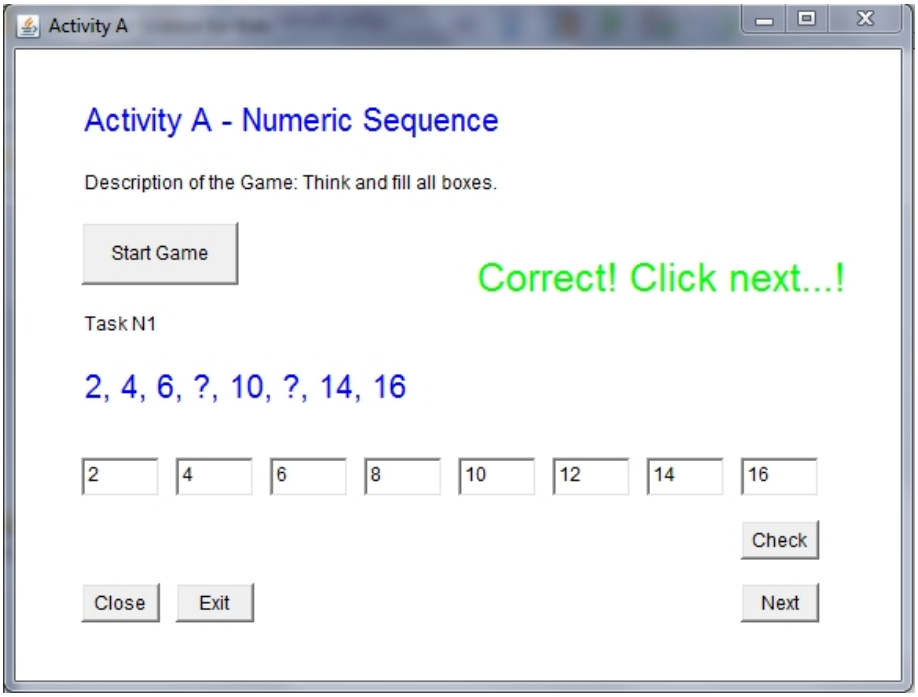


Button **Start Game**, takes first record of activities from file **ActivityA** and displayes on screen. If there are no activities, output message will ask student to either leave program or chose another activity.

```
void CreateArray() {
    tActivityA.createArray();
    ListA = tActivityA.ListA;
    k = tActivityA.k;  // Number of activities created by teacher in Teacher's Section A
}

void Start() { // The first method which beggins Student Section's functuality - Activity A
    CreateArray();
    if (k >= 0) {
        count = 1;
        task = ListA[0][0] + ", " + ListA[0][1] + ", " + ListA[0][2] + ", " +
               ListA[0][3] + ", " + ListA[0][4] + ", " + ListA[0][5] + ", "
               + ListA[0][6] + ", " + ListA[0][7];
        taskNum = "Task N" + count;

        dispose();
        new sActivityA();
    } else {
        outputString("No activity!\n Please finish or start other activity!");
    }
```

This class uses methods and variables from tActivityA.java class; tActivityA.createArray().

After filling empty boxes in, student presses **Check** - which reads numbers and compares them with array's elements with indexes 8-15. If they are all equal-**Correct** message is displayed if not-**Incorrect**. If any box is left blank message reminds students to fill all boxes.




```
if (c1.equals(ListA[count - 1][8]) && c2.equals(ListA[count - 1][9])
      && c3.equals(ListA[count - 1][10]) && c4.equals(ListA[count - 1][11])
    && c5.equals(ListA[count - 1][12]) && c6.equals(ListA[count - 1][13])
   && c7.equals(ListA[count - 1][14]) && c8.equals(ListA[count - 1][15])) {
    CI.setText("Correct! Click next...!");
    CI.setForeground(Color.GREEN);
} else {
    CI.setText("Incorrect! Try again...!");
    CI.setForeground(Color.RED);
}
```

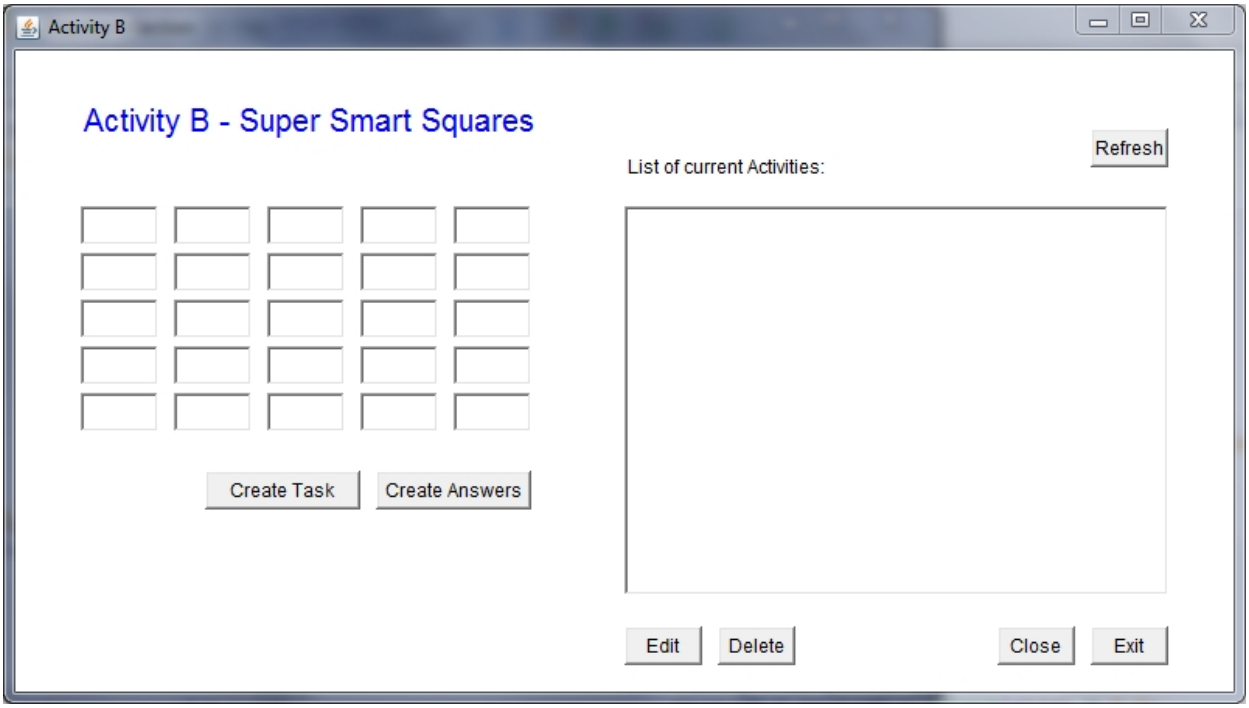
After pressing button **Next**, program checks if label said **Correct** or **Incorrect.** If it was **Correct**, new activity is displayed, if not message **"Please, try again!"**. If no activities are left Game is Over.

```
void Next() {
    String ci = CI.getText();
    if (ci.equals("Correct! Click next...!")) {

        task = ListA[count][0] + ", " + ListA[count][1] + ", " + ListA[count][2]
               + ", " + ListA[count][3] + ", " + ListA[count][4] + ", "
               + ListA[count][5] + ", " + ListA[count][6] + ", " + ListA[count][7];
        count = count + 1;
        taskNum = "Task N" + count;
        dispose();
        if (count <= k) {
            new sActivityA();
        } else {
            outputString("Game Over!\n Well Done!");
        }
    } else {
        outputString("Please, try again!");
    }
```

Variable **count** counts tasks and compares with k (number of tasks created) to finish game.

Button **Close** closes window and button **Exit** ends program.

Activity B – Teacher's Section – tActivityB.java



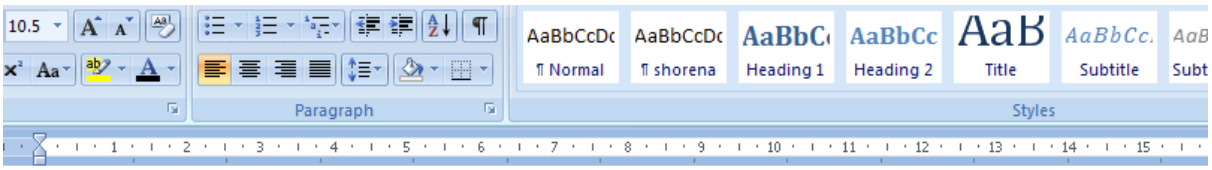
After filling up boxes with numbers and "**?**" teacher presses button **Create Task**-which gets text from every 25 boxes and creates **arrayT** (task array).

```
// variables declaration
String[][] arrayT = new String[5][5]; //for temporary tasks to create them
String[][] arrayA = new String[5][5]; // for temporary answers to create them
int order = 0;  //till task is created; after creating the task, order=1;

// methods section
public void CreateT() {
    if (order == 1) {
        outputString("Task is alrady created; \nPlease, create answers!");
    } else {
        try {
            RandomAccessFile tActivityB = new RandomAccessFile("ActivityB.txt", "rw");
            arrayT[0][0] = T1.getText();
            arrayT[0][1] = T2.getText();
            arrayT[0][2] = T3.getText();
            arrayT[0][3] = T4.getText();
            arrayT[0][4] = T5.getText();
            arrayT[1][0] = T6.getText();
```

```
131                     for (int i = 0; i < 5; i++) {
132                         for (int j = 0; j < 5; j++) {
133                             problem = problem + arrayT[i][j] + " ";
134                         }
```

After creating array, problems are written in random access file **ActivityB**. If any box is left empty warning message is displayed**.** After pressing button **Create Task**, already filled up boxes are not getting empty (not to type 25 numbers over again).

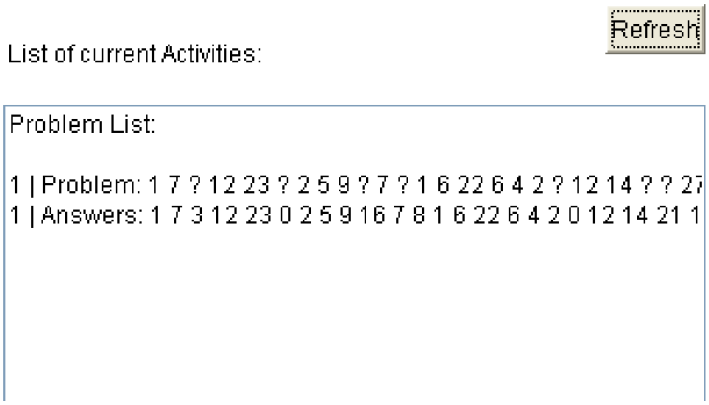
Button **Create Answers** – creates answers with same principal as tasks were created. Answers are written under corresponding tasks. If teacher presses **Create Answers** before **Create Tasks** message will warn to create tasks firstly.




```
Problem: 1 7 ? 12 23 ? 2 5 9 ? 7 ? 1 6 22 6 4 2 ? 12 14 ? ? 27 ?
Answers: 1 7 3 12 23 0 2 5 9 16 7 8 1 6 22 6 4 2 0 12 14 21 11 27 4
```

Button **Refresh**-reads all activities from file and displays in list. If list is already displayed, **Refresh** removes all activities and then read from file.

```
public void Refresh() {
    List.removeAll();
    count = 0;
    List.add("Problem List: ");
    List.add("");

    try {
        RandomAccessFile tActivityB = new RandomAccessFile("ActivityB.txt", "rw");
        while (tActivityB.getFilePointer() != tActivityB.length()) {
            String problem = tActivityB.readLine();
            String answer = tActivityB.readLine();
            count++;
            List.add(count + " | " + problem);
            List.add(count + " | " + answer);

        }
    } catch (IOException e) {
        e.getMessage();
    }
}
```



After pressing button **Edit**, firstly program determines wheatear selected item is problem or answer.

```
314             if (item.substring(4, 11).equals("Problem")) {
315                 index = 0; //selected item is a problem
316             } else {
317                 index = 1; //selected item is an answer
318             }
```

No matter which one teacher will select he/she still has to edit firstly problem and then answer. Numbers are taken by indexes and put in corresponding box.

```
322             int index1 = 0;
323             int index2 = paArray[itemN][index].indexOf(" ");
324             T1.setText(paArray[itemN][index].substring(0, index2));
```

Same is with other 25 numbers (only number of T is changing).

For Edit and Delete paArray is created – data is taken from file:

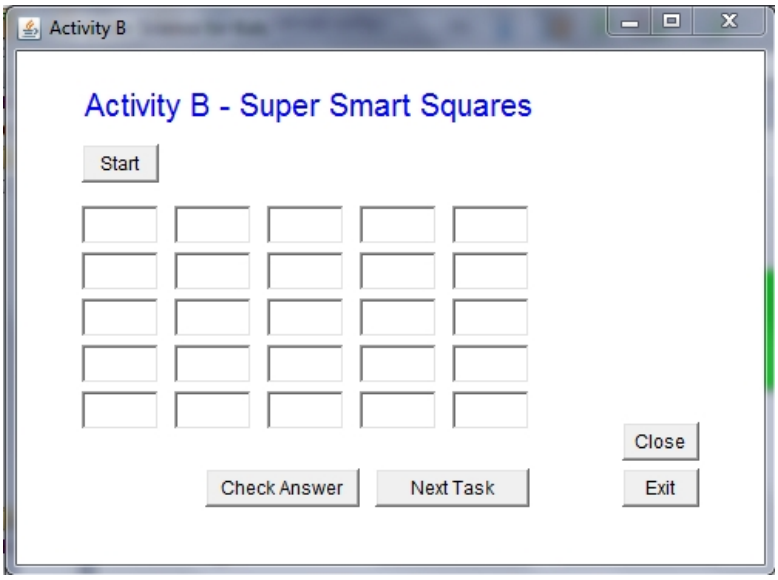
```
static String[][] paArray = new String[30][2];
static int countPA = 0;
```

```
static void createArray() {
    countPA = 0;
    String line1 = "";
    String line2 = "";
    try {
        RandomAccessFile tActivityB = new RandomAccessFile("ActivityB.txt", "rw");
        tActivityB.seek(0);
        while (tActivityB.length() != tActivityB.getFilePointer()) {
            line1 = tActivityB.readLine();
            line2 = tActivityB.readLine();
            paArray[countPA][0] = line1.substring(9);
            paArray[countPA][1] = line2.substring(9);
            countPA++;
        }
```

After pressing button **Delete,** if nothing is selected message will say, **"Select item!"** If item is selected, method **createArray()** creates array of activities. In **Delete()** every item is written in list if it does not equal to selected item.

```
try {
    RandomAccessFile tActivityB = new RandomAccessFile("ActivityB.txt", "rw");
    tActivityB.setLength(0);
    for (int i = 0; i < countPA; i++) {
        if (!paArray[i][0].equals(item.substring(13)) &&
                !paArray[i][1].equals(item.substring(13))) {
            tActivityB.writeBytes("Problem: " + paArray[i][0] + "\n");
            tActivityB.writeBytes("Answers: " + paArray[i][1] + "\n");
        }
```

Activity B – Student's Section

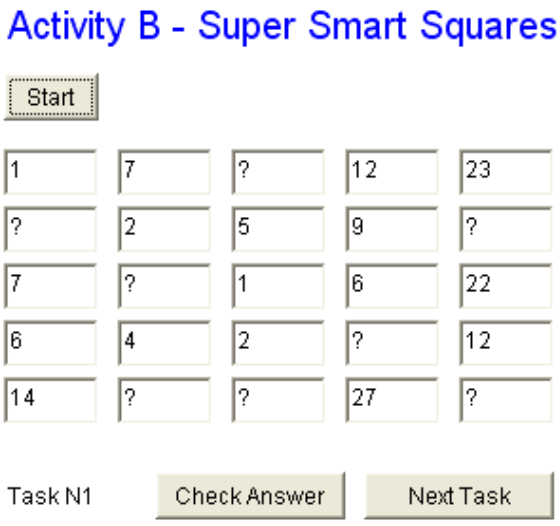


After pressing button **Start**, **Start()** method operates and calls method **taskDisplay(). taskDisplay()** writes each number in corresponding box.

```
72          int index1 = 0;
73          int index2 = taskLine.indexOf(" ");
74          T1.setText(taskLine.substring(0, index2));
```

(Setting text is same for all 25 boxes).

With displaying tasks, program makes label of which number of task is displayed on screen.



After pressing button **Check Answer, Check()** gets all numbers in one string answer. To check, program compares answer to numbers written in array. If every number coincides than corresponding message is displayed.

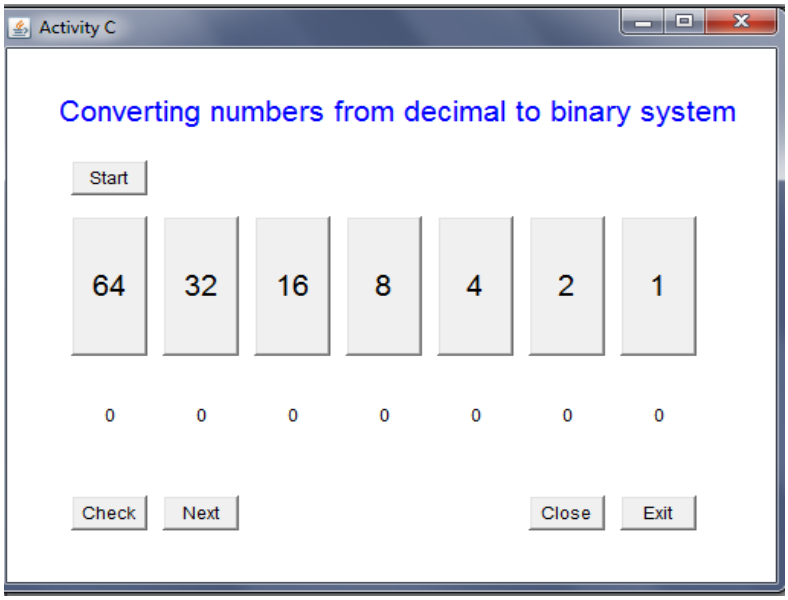
```
public void Check() {
    String answerLine = tActivityB.paArray[count][1];
    answer = T1.getText() + " " + T2.getText() + " " + T3.getText() + " " +
            T4.getText() + " " + T5.getText() + " " + T6.getText() + " " +
            T7.getText() + " " + T8.getText() + " " + T9.getText() + " " +
            T10.getText() + " " + T11.getText() + " " + T12.getText() + " " +
            T13.getText() + " " + T14.getText() + " " + T15.getText() + " " +
            T16.getText() + " " + T17.getText() + " " + T18.getText() + " " +
            T19.getText() + " " + T20.getText() + " " + T21.getText() + " " +
            T22.getText() + " " + T23.getText() + " " + T24.getText() + " " +
            T25.getText() + " ";
    if (answer.equals(tActivityB.paArray[count][1])) {
        CI.setText("Correct! Click next...!");
        CI.setForeground(Color.GREEN);
    } else {
        CI.setText("Incorrect! Try again...!");
        CI.setForeground(Color.RED);
```

After pressing button **Next**, program checks if label after button **Check** was **Correct** or **Incorrect.** If it was **Correct** new activity is displayed, if not then message **"Please, try again!"** is thrown. If no activities left in list - **"Game Over! Well Done!"**

```
177     public void Next() {
178
179         String ci = CI.getText();
180         CI.setText("");
181         if (ci.equals("Correct! Click next...!")) {
182             count++;
183             if (count < tActivityB.countPA) {
184                 taskLine = tActivityB.paArray[count][0];
185                 taskDisplay(taskLine);
186                 taskNum.setText("Task N" + (count + 1));
187             } else {
188                 outputString("Game Over!\n Well Done!");
189             }
190         } else {
191             outputString("Please, try again!");
192         }
193     }
194 }
```

Activity C

Activity_C is generated and checked by the program.



After pressing button **Start**, number from 0 to 99 are randomly chosen and displayed on screen.

```
89      void Start() {
90          int number = (int) (Math.random() * 100);
91          Number.setText(Integer.toString(number));
92      }
```

As the maximum number to represent is 99 (client's request) the maximum power of 2 is $2^6=64$;

After pressing any button with power of 2, corresponding label changes either with 1 or 0 (accordingly to how many times you press it). In case of mistake student is able to change his/her decision.

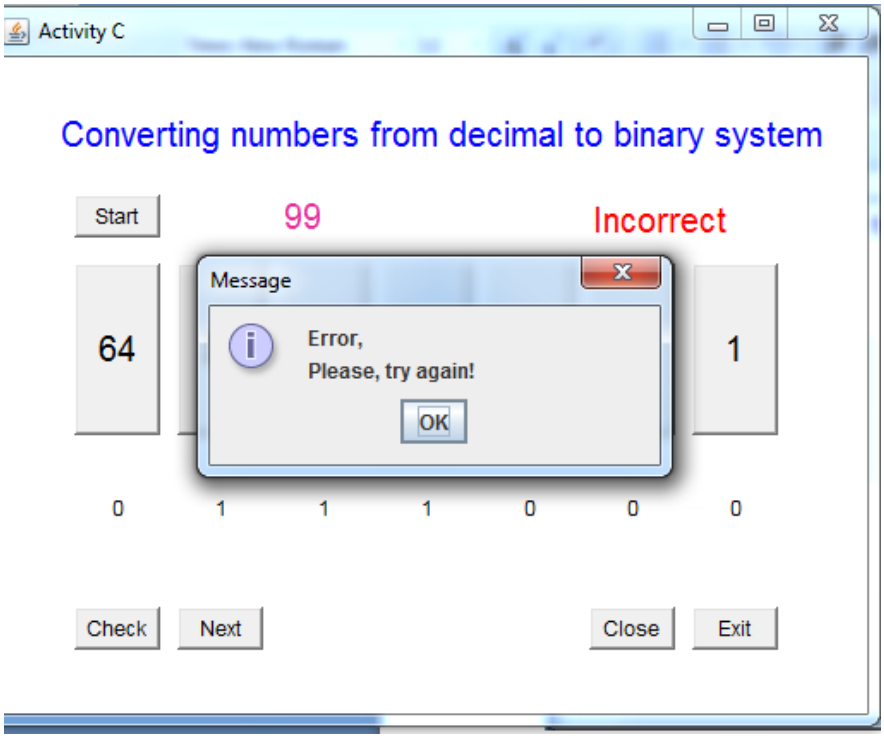
```
104      void B1() {
105          String b1 = L1.getText();
106          if (b1.equals("0")) {
107              L1.setText("1");
108          } else {
109              L1.setText("0");
110          }
111      }
```

There are 7 buttons with the same code;

After pressing button **Check** program reads all labels and multiples each of them by corresponding power of 2. Then program sum them up and if given number and result are same, program displays “Correct” - if not “Incorrect”.

```
void Check() {
    int c1 = Integer.parseInt(L1.getText());
    int c2 = Integer.parseInt(L2.getText());
    int c3 = Integer.parseInt(L3.getText());
    int c4 = Integer.parseInt(L4.getText());
    int c5 = Integer.parseInt(L5.getText());
    int c6 = Integer.parseInt(L6.getText());
    int c7 = Integer.parseInt(L7.getText());

    int answer = (int) (c1 * Math.pow(2, 6) + c2 * Math.pow(2, 5) +
          c3 * Math.pow(2, 4) + c4 * Math.pow(2, 3) +
          c5 * Math.pow(2, 2) + c6 * Math.pow(2, 1) + c7 * Math.pow(2, 0));

    if (answer == Integer.parseInt(Number.getText())) {
        CI.setForeground(Color.getHSBColor(0.3f, 1f, 0.7f));
        CI.setText("Correct"); // writes in green
    } else {
        CI.setForeground(Color.getHSBColor(0f, 1f, 1f));
        CI.setText("Incorrect");  //writes in red
    }
}
```

When button **Next** is pressed, program checks if label after pressing button **Check** said “correct” or “incorrect”. If label was “correct” than **Next** does same as did button **Start**; chooses another random number. Also **Next** is deleting previously written number and returns to original state;



If student's answer was "incorrect", Error message is displayed to give chance to correct answer. Student can finish with Activity_C any time he/she wishes.

Words = 1074