API, IDE, ΤΕΚΜΗΡΙΩΣΗ ΚΑΙ ΔΙΑΝΟΜΗ ΛΟΓΙΣΜΙΚΟΥ

Δημήτρης Κ. Ιακωβίδης

Στόχος αυτής της ενότητας είναι η επισκόπηση των εννοιών που σχετίζονται με τις διεπαφές προγραμματισμού εφαρμογών (Application Programming Interfaces, APIs), την εισαγωγή στην ανάπτυξη τεκμηριωμένου λογισμικού και την οργάνωση των κλάσεων πακέτα (packages) ή αυτόνομα συμπαγή αρχεία διανομής, τύπου jar. Περιγράφεται επίσης η χρήση ολοκληρωμένων περιβαλλόντων ανάπτυξης λογισμικού (Integrated Development Environments, IDEs).

8.1 Χρήση πακέτων

Από τα πρώτα στοιχεία που μαθαίνουμε στον προγραμματισμό με Java είναι να χρησιμοποιούμε πακέτα. **Τα πακέτα είναι οργανωμένες συλλογές (βιβλιοθήκες) κλάσεων**. Για να μπορούμε να χρησιμοποιούμε μια κλάση από ένα πακέτο, πρέπει να έχουμε προσθέσει στην αρχή της κλάσης μας την πρόταση **import** ακολουθούμενη από το όνομα του πακέτου, π.χ.

import java.util.Random;

Η πρόταση αυτή μας επιτρέπει να χρησιμοποιούμε στην κλάση μας την κλάση Random του πακέτου **java.util**, ενώ η παρακάτω πρόταση:

import java.util.*;

η οποία κάνει χρήση του **χαρακτήρα μπαλαντέρ** '*', μας επιτρέπει να χρησιμοποιούμε οποιαδήποτε κλάση του ίδιου πακέτου.

Οι πιο έμπειροι προγραμματιστές επιλέγουν συνήθως τον πρώτο τρόπο, επαναλαμβάνοντας την πρόταση import όσες φορές είναι απαραίτητο για κάθε κλάση που επιθυμούν να εισάγουν. Τούτο, αν και είναι επίπονο στη συγγραφή του προγράμματος, επιτρέπει να γνωρίζουμε ποιες εξωτερικές κλάσεις χρησιμοποιούνται από την κλάση μας. Όσες φορές και να καλείται η πρόταση import δεν επηρεάζει τις επιδόσεις του προγράμματός μας. Στο Σχ.1 δίνεται παράδειγμα χρήσης γνωστών κλάσεων από το πακέτο java.util για την κατασκευή ενός μικρού παιχνιδιού, όπου ο χρήστης κερδίζει αν μαντέψει σωστά έναν αριθμό, ανάμεσα στο 0 και στο 10. Το παιχνίδι επαναλαμβάνεται 10 φορές.

```
1    import java.util.Random;
2    import java.util.Scanner;
3
4    class GuessGame
5    {
6          public static void main(String[] args)
7          {
8                int score = 0;
9                for (int i = 0; i < 10; i++)
10               {
11                     // Ask for a number between 0-10
12                     Scanner s = new Scanner(System.in);
13                     System.out.print("Number? ");
14                     int input = s.nextInt();
15
16                     // Generate a random number between 0-10
17                     Random r = new Random();
18                     int guess = r.nextInt(10);
19
20                     // If the numbers match you win!
21                     if (input == guess)
22                     {
23                           score++;
24                           System.out.println("You won!");
25                     } else
26                     {
27                           System.out.println("You lost.");
28                     }
29               }
30               System.out.println("Your score is " + score +
31                     "/10.");
32         }
33   }
```

Σχήμα 1. Χρήση κλάσεων του πακέτου **java.util** για την κατασκευή ενός απλού παιχνιδιού.

8.2 Κατασκευή τεκμηρίωσης API

Η Java συνοδεύεται από μια χρήσιμη εφαρμογή που ονομάζεται **javadoc**. Η εφαρμογή αυτή κατασκευάζει αυτόματα τεκμηρίωση στις κλάσεις μας, με μορφή ίδια με εκείνη της τεκμηρίωσης του Java API. Αυτό το επιτυγχάνει "διαβάζοντας" τα σχόλια που έχουμε εισάγει στον κώδικα με ειδικό τρόπο. Στα σχόλια που εισάγουμε χρησιμοποιούμε κάποια σύμβολα και ετικέτες (tags) που σημειώνονται με '@', τα οποία σηματοδοτούν το μετασχηματισμό των σχολίων σε μορφή τεκμηρίωσης. Στον Πίνακα 1 παρατίθεται ένα υποσύνολο από τα πιο συχνά χρησιμοποιούμενα από αυτά.

Στο Σχ.2 παρουσιάζεται η χρήση των συμβόλων του Πίνακα 1 για την τεκμηρίωση ενός προγράμματος με τη χρήση του javadoc. Για να παραχθεί απλά η τεκμηρίωση σε ένα νέο φάκελο με όνομα **documentation** πρέπει να γράψετε στη γραμμή εντολών την εξής πρόταση:

javadoc -d documentation GuessGame.java

όπου η παράμετρος –d δηλώνει το όνομα του φακέλου. Για να προστεθούν οι πληροφορίες αναφορικά με το συγγραφέα και την έκδοση του προγράμματος πρέπει να προσθέσετε τις παραμέτρους –author και –version αντίστοιχα, π.χ.,

javadoc -author -version -d documentation GuessGame.java

Για επιπλέον παραμέτρους που υποστηρίζει το javadoc, απλά πληκτρολογείστε javadoc στη γραμμή εντολών και διαβάστε το βοηθητικό κείμενο που εμφανίζεται.

Πίνακας 1. Σύμβολα για την αυτόματη τεκμηρίωση με τη χρήση του **javadoc**.

Σύμβολα	**Χρήση**
/**	Έναρξη σχολίων που αναγνωρίζει το javadoc για την αυτόματη κατασκευή τεκμηρίωσης. Χρησιμοποιείται αντί του τυπικού '/*'.
@param	Μετά την ετικέτα αυτή ακολουθεί το όνομα και η περιγραφή μιας παραμέτρου μιας μεθόδου.
@return	Μετά την ετικέτα αυτή ακολουθεί η περιγραφή αναφορικά με το τι επιστρέφεται από μια μέθοδο.
@author	Ακολουθεί το όνομα του συγγραφέα (προγραμματιστή).
@version	Ακολουθεί η έκδοση του λογισμικού.

```
1     import java.util.Random;
2     import java.util.Scanner;
3
4     /**
5     GuessGame: A number guessing game.
6     @author Dimitris Iakovidis
7     @version 0.1
8     */
9     public class GuessGame
10    {
11          public static String
12                MESSAGE_WON       = "You won!\n",
```

```
13                MESSAGE_LOST      = "You lost\n",
14                MESSAGE_SCORE     = "Your score is ",
15                MESSAGE_QUESTION  = "Number? ";
16
17          private int maxIterations,
18                      maxRandom,
19                      iteration,
20                      score,
21                      input;
22
23          /**
24          Constructs a guess game given a set of parameters
25          @param maxIterations The number of iterations to run
26                 the game
27          @param maxRandom The maximum random number to guess
28          */
29          public GuessGame(int maxIterations, int maxRandom)
30          {
31                score = 0;
32                this.maxIterations = maxIterations;
33                this.maxRandom = maxRandom;
34          }
35
36          /**
37          Constructs a guess game using default parameters
38          */
39          public GuessGame()
40          {
41                this(10, 10);
42          }
43
44          /**
45          Increases the score by one
46          */
47          public void increaseScore()
48          {
49                score++;
50          }
51
52          /**
53          Returns the current score
54          @return The current score
55          */
56          public int getScore()
57          {
58                return score;
59          }
60
61          /**
62          Returns the number of iterations that the game will
63          run
64          @return The number of iterations that the game will
65          run
66          */
67          public int getMaxIterations()
68          {
69                return maxIterations;
70          }
71
72          /**
```

```
73          Returns a random number between 1 and a maximum value
74          maxRandom
75          @return A random number between 1 and a maximum value
76          maxRandom
77          */
78          public int generateRandom()
79          {
80                Random r = new Random();
81                int guess = r.nextInt(maxRandom);
82                return guess;
83          }
84
85          /**
86          Asks the user a number from the command line
87          @return The number entered by the user
88          */
89          public int read()
90          {
91                Scanner s = new Scanner(System.in);
92                System.out.print(MESSAGE_QUESTION);
93                int input = s.nextInt();
94                return input;
95          }
96
97          /**
98          Outputs a string to the command line
99          @param s The string to be displayed in the output
100         */
101         public void write(String s)
102         {
103               System.out.print(s);
104         }
105
106         /**
107         Initiates a single question guess game play
108         */
109         public void playSingleQuestionGame()
110         {
111               for (int i = 0; i < maxIterations; i++)
112               {
113                     int input = read();
114                     int guess = generateRandom();
115
116                     // If the numbers match you win!
117                     if (input == guess)
118                     {
119                           increaseScore();
120                           write(MESSAGE_WON);
121                     } else
122                     {
123                           write(MESSAGE_LOST);
124                     }
125               }
126               write(MESSAGE_SCORE + score + "/" +
127                     maxIterations + ".");
128         }
129
130         /**
131         Executes a test game
132         */
```

```
133         public static void main(String[] args)
134         {
135               GuessGame x = new GuessGame();
136               x.playSingleQuestionGame();
137         }
138   }
```

Σχήμα 2. Μεταγραφή του παιχνιδιού του Σχ.1 ώστε να γίνει αυτόνομο και επαναχρησιμοποιήσιμο αντικειεμενοστρεφώς. Περιέχει επίσης σχόλια κατάλληλα για την αυτόματη παραγωγή τεκμηρίωσης.

generateRandom

`public int generateRandom()`

Returns a random number between 1 and a maximum value maxRandom

Returns:

A random number between 1 and a maximum value maxRandom

read

`public int read()`

Asks the user a number from the command line

Returns:

The number entered by the user

write

`public void write(java.lang.String s)`

Outputs a string to the command line

Parameters:

s - The string to be displayed in the output

playSingleQuestionGame

`public void playSingleQuestionGame()`

Initiates a single question guess game play

Σχήμα 3. Παράδειγμα του html αρχείου που παράγει το javadoc από τα σχόλια του κώδικα του Σχ. 2.

Παρατηρήστε ότι ο κώδικας του Σχ.2 υλοποιεί ότι ακριβώς και ο κώδικας του Σχ.1. Όμως ο κώδικας του Σχ.1 δε μπορεί να χρησιμοποιηθεί παρά μόνο για να εκτελεί το συγκεκριμένο παιχνίδι μαντείας αριθμών. **Ο κώδικας του Σχ.2 είναι αντικειμενοστρεφώς σχεδιασμένος ώστε να μπορεί κανείς να τον χρησιμοποιήσει για την υλοποίηση και άλλων παρόμοιων παιχνιδιών ή να τον επεκτείνει για εκτέλεση όχι μόνο από τη γραμμή εντολών.**

8.3 Κατασκευή πακέτων

Ένα πακέτο κατασκευάζεται όπως και μια οργάνωση αρχείων σε φακέλους. Έστω ότι επιθυμούμε να κατασκευάσουμε ένα πακέτο με κλάσεις που αναπαριστούν διαφορετικά ζώα (animals), της γης (earth), του ουρανού (sky) και της θάλασσας (sea). Τα βήματα που πρέπει να ακολουθήσουμε είναι τα εξής:

1. Κατασκευάζουμε στην τρέχουσα διαδρομή του συστήματος αρχείων μας τους εξής φακέλους:

animals

animals/earth

animals/sky

animals/sea

και στους φακέλους earth, sky και sea τοποθετούμε τις κλάσεις με τα αντίστοιχα ζώα, π.χ. στον earth τις κλάσεις Cat.java, Dog.java, στον sky την κλάση eagle.java στον sea την κλάση dolphin.java.

2. Στο αρχείο κάθε κλάσης (πριν από τη λέξη class) τοποθετούμε μια πρόταση που περιγράφει σε ποιο πακέτο ανήκει, π.χ., στο Cat.java, τοποθετούμε την πρόταση (Σχ. 4):

package animals.earth;

3. Οι κλάσεις που επιθυμούμε να προσπελαύνονται εξωτερικά από τους χρήστες του πακέτου πρέπει να είναι δηλωμένες ως **public**.

4. Μεταγλωττίζουμε κάθε κλάση του πακέτου μας για να παραχθούν τα αντίστοιχα αρχεία τύπου **class**.

5. Κατασκευάζουμε ένα συμπαγές αρχείο **animals.jar** ως εξής:

jar -cvf animals.jar animals

Για να μπορούμε να εισάγουμε την κλάση αυτή στα προγράμματά μας με

import animals.earth.Cat;

θα πρέπει να προσθέσουμε τη διαδρομή προς το πακέτο **animals.jar** στη μεταβλητή περιβάλλοντος **classpath**, π.χ.

SET CLASSPATH=.;C:\Program Files\Java\jdk1.7.0_40\LIB\TOOLS.JAR;C:\myfolder\animals.jar

ή εφόσον όπως στο παραπάνω παράδειγμα υπάρχει ο φάκελος '.' στην αρχή της πρότασης SET, να διασφαλίσουμε ότι **το πακέτο animals.jar βρίσκεται στον ίδιο φάκελο με το αρχείο java του προγράμματος μας στο οποίο τη χρησιμοποιούμε**.

```
1    package animals.earth;
2
3    public class Cat
4    {
5          public String name;
6
7          // Enables a cat to speak
8          public void speak()
9          {
10               System.out.println(name + " says Miaou");
11         }
12
13         // Cat becomes executable
14         public static void main(String[] args)
15         {
16               // Our cat's name is Goldy
17               Cat g = new Cat();
18               g.name = "Goldy";
19               g.speak();
20         }
21   }
```

Σχήμα 4. Παράδειγμα κλάσης για το πακέτο **animals.earth**.

8.4 Δημιουργία εκτελέσιμης διανομής

Για να είναι εκτελέσιμο ένα πακέτο πρέπει να περιέχει μια τουλάχιστον κλάση που έχει μέθοδο main. Θα πρέπει με κάποιο τρόπο να υποδείξουμε στο jar αρχείο ποιας κλάσης main επιθυμούμε να εκτελείται όταν κάνουμε δύο φορές κλικ στο όνομα του jar για να εκτελεσθεί. Αυτό γίνεται μέσω ενός αρχείου κειμένου που τοποθετείται μέσα στο jar. Αυτό γίνεται ως εξής:

1. Δημιουργούμε ένα νέο αρχείο με όνομα manifest.txt στον εξωτερικό φάκελο του πακέτου μας, π.χ. όχι μέσα στον φάκελο animals.
2. Αν το αρχείο με τη μέθοδο main που θέλουμε να εκτελείται είναι το Cat, τότε μέσα στο αρχείο γράφουμε την παρακάτω πρόταση και το αποθηκεύουμε:

   Main-Class: animals.earth.Cat

ΠΡΟΣΟΧΗ! Για να λειτουργήσει σωστά πρέπει α) να υπάρχει κενό ανάμεσα στην άνω-κάτω τελεία και τη λέξη "animals", β) να έχει πατηθεί enter στο τέλος της λέξης Cat. Διαφορετικά δε θα δημιουργείται σωστά το manifest.mf μέσα στο jar.

3. Κατασκευάζουμε το αρχείο jar της κλάσης μας ως εξής:

   jar -cvfm animals.jar manifest.txt animals

Το αποτέλεσμα αυτής της διαδικασίας θα είναι ένα πακέτο τύπου **jar** το οποίο θα εκτελείται με δύο κλικ, θα εκτελείται δηλαδή η μέθοδος main της κλάσης **Cat**. Αυτή είναι η συνήθης μορφή διανομής εφαρμογών Java. Από τη γραμμή εντολών, το εκτελέσιμο αρχείο jar εκτελείται με την πρόταση:

java –jar animals.jar

8.5 Χρήση IDE

Τα πιο δημοφιλή δημοσίως διαθέσιμα IDE για την υποβοηθούμενη συγγραφή λογισμικού σε Java είναι το **Netbeans** και το **Eclipse**. Βασικές λειτουργίες τους:

- Χρωματισμός του κώδικα για ευαναγνωσιμότητα.
- Συνεχής εκτέλεση του μεταγλωττιστή javac για τον εντοπισμό σφαλμάτων.
- Αυτόματη κλήση των javadoc και jar για τη δημιουργία τεκμηρίωσης και κατασκευή πακέτων για τη διανομή του λογισμικού μας.
- Αυτόματη αναζήτηση και μετονομασία κλάσεων, μεταβλητών, αλλαγή τμημάτων κώδικα κλπ (refactoring)

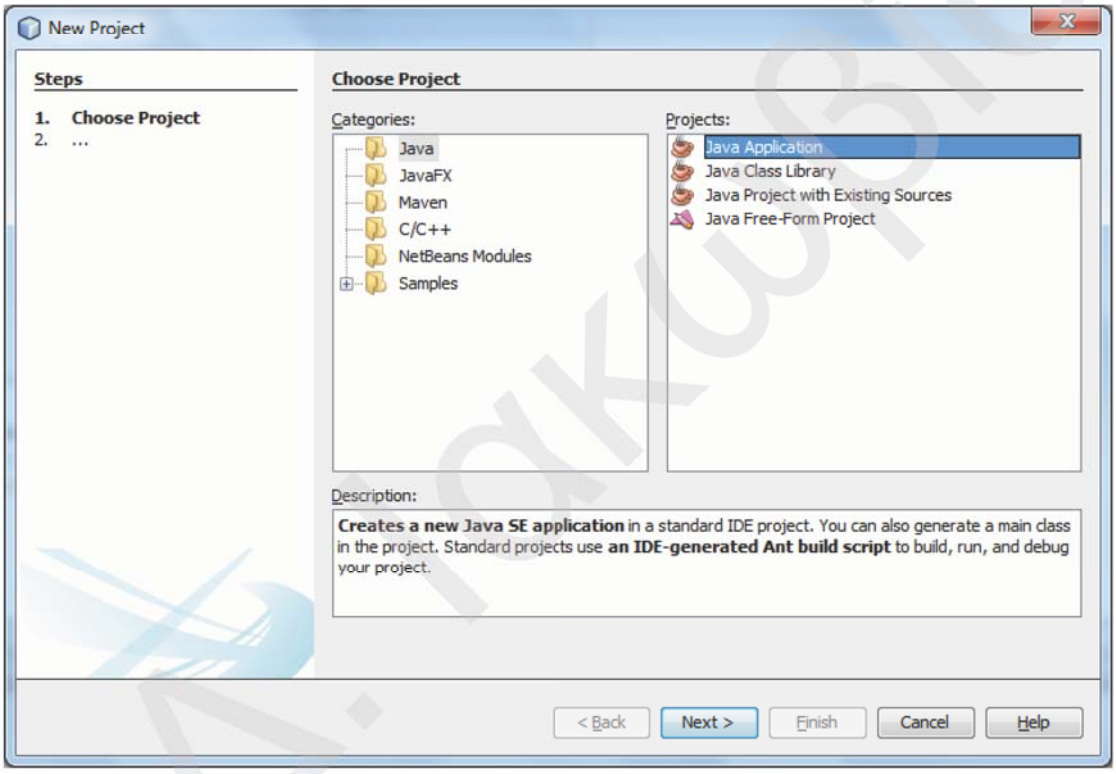
Πλεονέκτημα του Eclipse έναντι του Netbeans είναι η ευκολία στην κατασκευή εκτελέσιμου πακέτου στο οποίο μπορούν να ενσωματωθούν και οι εξωτερικές βιβλιοθήκες από τις οποίες εξαρτάται.

Και τα δύο IDE μας επιβάλουν αρχικά να δημιουργήσουμε ένα νέο **έργο (project)** για το πρόγραμμά μας. Αυτό μπορεί να περιλαμβάνει πηγαίο κώδικα (source code) και βιβλιοθήκες (libraries / packages) από τις οποίες θα εξαρτάται ο πηγαίος κώδικας μας. Ο προσδιορισμός αυτών των στοιχείων μπορεί να γίνει είτε στην αρχή ακολουθώντας τα βήματα του οδηγού είτε κατόπιν, χειροκίνητα.

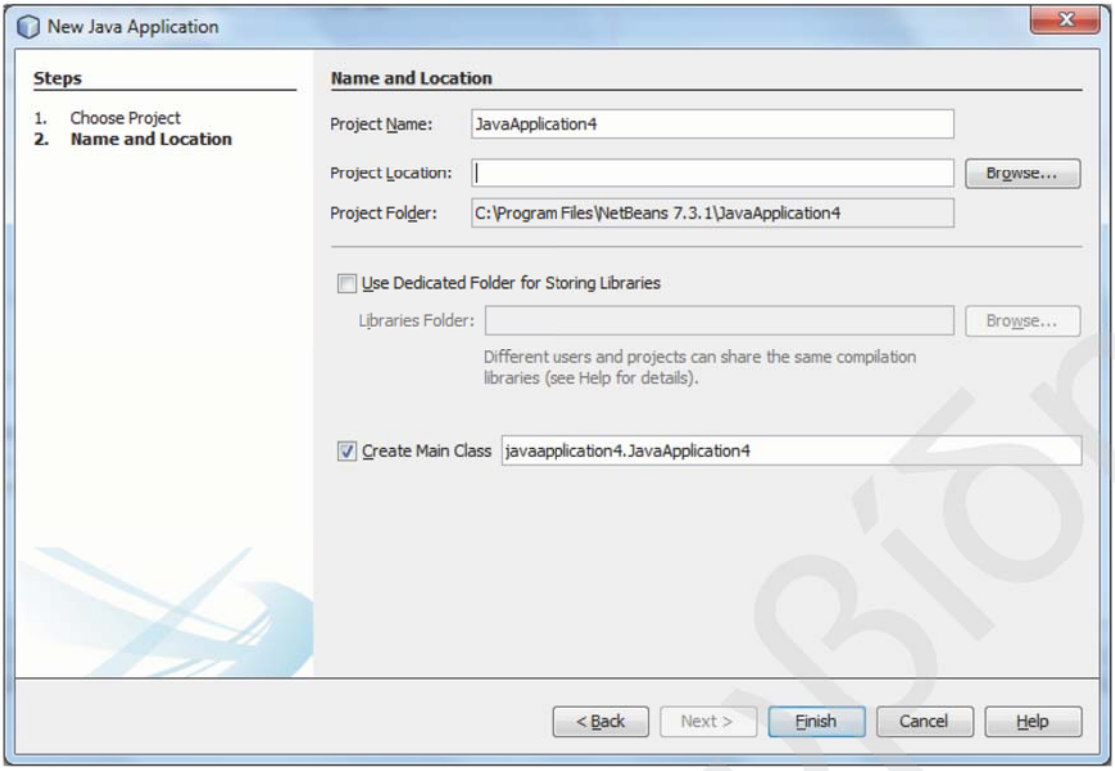
1. Για τη δημιουργία ενός έργου με το Netbeans ακολουθούμε τις επιλογές του μενού **File → New Project →**

2. Επιλέγουμε στο παράθυρο του Σχ.5 τι επιθυμούμε να δημιουργήσουμε. Αν π.χ. επιλέξουμε Java Application θα δημιουργηθεί ένα νέο έργο με έτοιμη τη μέθοδο main.

3. Σε επόμενο παράθυρο (Σχ.6) συμπληρώνουμε το όνομα του έργου μας, τις διαδρομές (paths) στα οποία επιθυμούμε να δημιουργηθεί, αν επιθυμούμε να δημιουργηθεί main και σε ποια κλάση (στο παράδειγμα η κλάση είναι η JavaApplication4 του πακέτου javaapplication4).



Σχήμα 5. Πρώτο παράθυρο του οδηγού δημιουργίας νέου έργου στο Netbeans IDE.



Σχήμα 6. Δεύτερο παράθυρο του οδηγού δημιουργίας νέου έργου στο Netbeans IDE.

```
1   /*
2    * To change this template, choose Tools | Templates
3    * and open the template in the editor.
4    */
5   package javaapplication4;
6
7   /**
8    *
9    * @author Dimitris
10   */
11  public class JavaApplication4 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18          System.out.println("test");
19      }
20  }
```

Σχήμα 7. Κώδικας που παράγεται αυτόματα από το Netbeans IDE.

Παρατηρείστε ότι στον κώδικα του Σχ.7 που παράγεται από το Netbeans IDE υπάρχουν:

α) σχόλια τεκμηρίωσης javadoc,

β) η πρόταση package, και

γ) μια μέθοδος main.

Για την παραγωγή του αρχείου της τεκμηρίωσης αρκεί να επιλέξουμε από το μενού **Run → Generate Javadoc**.

- Για να **μεταγλωττίσετε** το πρόγραμμά σας κάνετε κλικ στο εικονίδιο με το σφυράκι και το σκουπάκι.
- Για να **εκτελέσετε** το πρόγραμμά σας κάνετε κλικ στο εικονίδιο με το πράσινο δεξί βέλος.
- Τα σφάλματα υπογραμμίζονται και αριστερά εμφανίζεται ένα εικονίδιο μια λάμπα. Όταν μετακινήσετε το βελάκι σας επάνω από το εικονίδιο αυτό εμφανίζεται το αντίστοιχο μήνυμα σφάλματος, για να βοηθηθείτε στην επίλυσή του.

Ασκήσεις

1. Να κατασκευάσετε ένα παιχνίδι εικονικού λόττο, όπως στο Σχ.1. Οι χρήστες θα δίνουν 6 αριθμούς από το 1 έως το 49, και αν και οι 6 είναι σωστοί τότε και μόνο τότε κερδίζουν.

2. Να αξιοποιήσετε όλη ή μέρος της κλάσης GuessGame του Σχ. 3 για την κατασκευή μιας μεθόδου playLotto της ίδιας κλάσης η οποία θα υλοποιεί το παιχνίδι της άσκησης 1.

3. Να πακετάρετε την κλάση GuessGame σε ένα jar αρχείο.

4. Να χρησιμοποιήσετε το javadoc για την τεκμηρίωση της κλάσης GuessGame.

5. Να τροποποιήσετε το παιχνίδι της άσκησης 2 ώστε να ανταγωνίζεται τον εαυτό του. Δηλαδή ο υπολογιστής να κάνει προσπάθειες να βρει τους 6 αριθμούς. Άραγε πόσες φορές απαιτείται να δοκιμάσει ο υπολογιστής για να κερδίσει το λόττο;