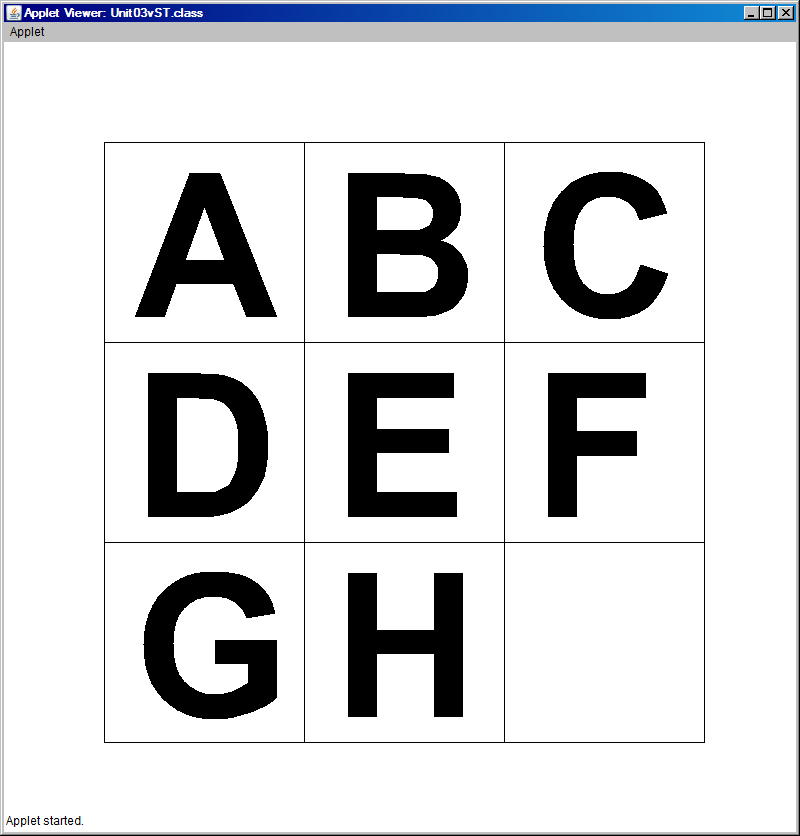
|  |  |  |
| --- | --- | --- |
| **Advanced Graphics Programming** | | **Unit 04 Lab** |
| **The Sliding Letter Puzzle** | **80, 100 & 110 Point Versions** | |
| **Assignment Purpose:**  The purpose of this program is to show knowledge of mouse interaction to play a popular traveling game of the past. Additionally, this lab assignment is designed to show how to alter a program from a "hard-coded" fixed size to a more general  "variable-size" game. | | |

Write a program, which plays the Letter Puzzle for a 3 X 3, 4 X 4 or 5 X 5 matrix size. The Letter Puzzle is a game based on an early travel game made out of plastic. Small squares with letters could be moved inside a larger square. There is always one open spot, which is used to move the next little square. The object of the game is to move the little letter squares in such a manner that the letters are alphabetically arranged. The student, starting file is a functional Letter Puzzle for a 3 X 3 game. It is the final stage of the provided program stages in the Unit03 puzzle game. The game randomly arranges the letters and looks like the display below.



The completed game arranges the letters in alphabetical order from left-to-right and top-to-bottom as the next display shows.



|  |
| --- |
| **Provided Student Program for Unit03vST**  // Unit04vST.java by Leon Schram 07-19-10  //  // This is the student starting file of the Unit04 lab assignment.  // The starting file is actually the Puzzle09.java stage, which is the  // finished puzzle game for a 3 X 3 matrix.  import java.awt.\*;  import java.applet.\*;  import java.util.Random;  import javax.swing.\*;  public class Unit04vST extends Applet  {    Rectangle r11,r12,r13,r21,r22,r23,r31,r32,r33;  boolean scramble[];  String matrix[][];  Random rnd;  int blankR;  int blankC;  int playLevel;        public void init()  {  r11 = new Rectangle(100,100,200,200);  r12 = new Rectangle(300,100,200,200);  r13 = new Rectangle(500,100,200,200);  r21 = new Rectangle(100,300,200,200);  r22 = new Rectangle(300,300,200,200);  r23 = new Rectangle(500,300,200,200);  r31 = new Rectangle(100,500,200,200);  r32 = new Rectangle(300,500,200,200);  r33 = new Rectangle(500,500,200,200);    matrix = new String[5][5];  scramble = new boolean[10];  for (int k = 1; k <=9; k++)  scramble[k] = false;  rnd = new Random();    for (int r = 0; r <= 4; r++)  for (int c = 0; c <= 4; c++)  matrix[r][c] = "#";    for (int r = 1; r <= 3; r++)  for (int c = 1; c <= 3; c++)  {  matrix[r][c] = getLetter();  if (matrix[r][c].equals("I"))  {  blankR = r;  blankC = c;  }  }  }      public String getLetter()  {  String letter = "";  boolean Done = false;  while(!Done)  {  int rndNum = rnd.nextInt(9) + 1;  if (scramble[rndNum] == false)  {  letter = String.valueOf((char) (rndNum+64));  scramble[rndNum] = true;  Done = true;  }  }  return letter;  }      public void paint(Graphics g)  {  drawGrid(g);  drawLetter(g,matrix[1][1],100,100);  drawLetter(g,matrix[1][2],300,100);  drawLetter(g,matrix[1][3],500,100);  drawLetter(g,matrix[2][1],100,300);  drawLetter(g,matrix[2][2],300,300);  drawLetter(g,matrix[2][3],500,300);  drawLetter(g,matrix[3][1],100,500);  drawLetter(g,matrix[3][2],300,500);  drawLetter(g,matrix[3][3],500,500);  }        public void drawGrid(Graphics g)  {  g.drawRect(100,100,600,600);  g.drawLine(300,100,300,700);  g.drawLine(500,100,500,700);  g.drawLine(100,300,700,300);  g.drawLine(100,500,700,500);  }      public void drawLetter(Graphics g, String letter, int x, int y)  {  int offSetX = x + 30;  int offSetY = y + 175;  g.setFont(new Font("Arial",Font.BOLD,200));  if (letter.equals("I"))  {  g.setColor(Color.white);  g.fillRect(x+1,y+1,198,198);  }  else  {  g.setColor(Color.black);  g.drawString(letter,offSetX,offSetY);  }  }      public boolean mouseDown(Event e, int x, int y)  {  if(r11.inside(x,y) && okSquare(1,1))  swap(1,1);  else if(r12.inside(x,y) && okSquare(1,2))  swap(1,2);  else if(r13.inside(x,y) && okSquare(1,3))  swap(1,3);  else if(r21.inside(x,y) && okSquare(2,1))  swap(2,1);  else if(r22.inside(x,y) && okSquare(2,2))  swap(2,2);  else if(r23.inside(x,y) && okSquare(2,3))  swap(2,3);  else if(r31.inside(x,y) && okSquare(3,1))  swap(3,1);  else if(r32.inside(x,y) && okSquare(3,2))  swap(3,2);  else if(r33.inside(x,y) && okSquare(3,3))  swap(3,3);  return true;  }      public boolean okSquare(int r, int c)  {  boolean temp = false;  if (matrix[r-1][c].equals("I"))  temp = true;  else if (matrix[r+1][c].equals("I"))  temp = true;  else if (matrix[r][c-1].equals("I"))  temp = true;  else if (matrix[r][c+1].equals("I"))  temp = true;  return temp;  }        public void swap(int r, int c)  {  matrix[blankR][blankC] = matrix[r][c];  matrix[r][c] = "I";  blankR = r;  blankC = c;  repaint();  }      public void update(Graphics g)  {  paint(g);  }      } |

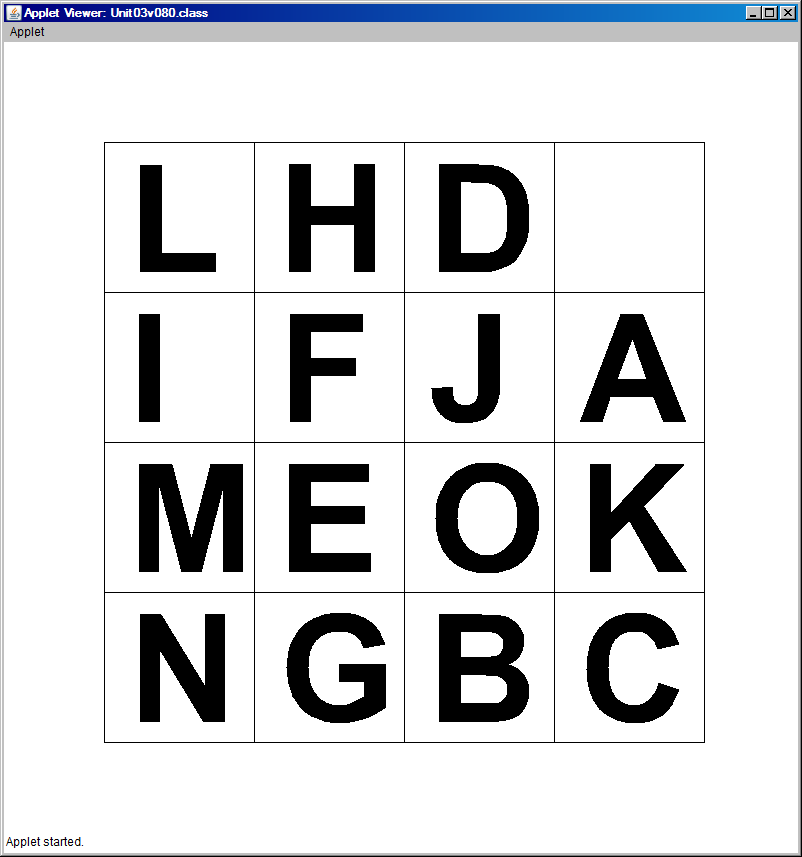
**80-Point Version**

The 80-point version of the program alters the starting file to allow a game using a 4 X 4 matrix. It is important to realize that it is not sufficient to create a game that looks like the display below. Such a game could be created by adding and altering the 3 X 3 game to become a 4 X 4 matrix. The starting file for the 3 X 3 game is written in a very inefficient manner. For instance, the nine **Rectangle** objects are created in nine separate statements as follows:

|  |
| --- |
| **r11 = new Rectangle(100,100,200,200);**    **r12 = new Rectangle(300,100,200,200);**    **r13 = new Rectangle(500,100,200,200);**    **r21 = new Rectangle(100,300,200,200);**    **r22 = new Rectangle(300,300,200,200);**    **r23 = new Rectangle(500,300,200,200);**    **r31 = new Rectangle(100,500,200,200);**    **r32 = new Rectangle(300,500,200,200);**    **r33 = new Rectangle(500,500,200,200);** |

For the 80-point version, it is not allowed to add additional objects in the same manner for a total of 16 objects. Your program needs to use loops where-ever appropriate rather than a series of individual program statements.

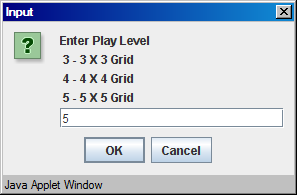
**Unit04 Lab Version-80 Point Required Execution**

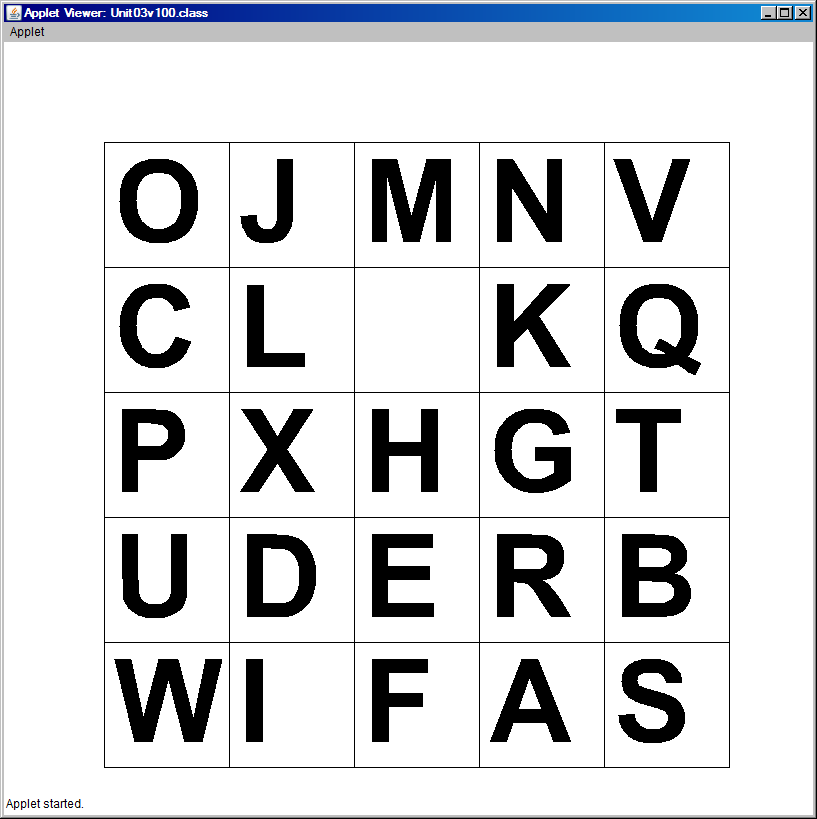


**100-Point Version**

For the 100-point version the game needs to be altered to include the 5 X 5 puzzle. However, this version is not simply a 5 X 5 game. The game now allows the user to select a 3 X 3, 4 X 4 or 5 X 5 puzzle. This means that many of the hard-coded constant values that were used in the starting version and the 80-point version must be changed to variables. The program starts by asking input from the player to select one of three play levels.

**Unit04 Lab Version-100 Point Required Execution**





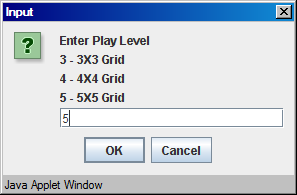
**110-Point Version**

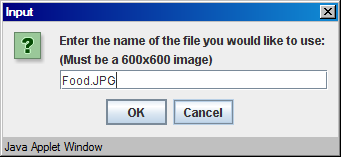
For the 110-Point extra credit version the Letter Puzzle changes and becomes an Image puzzle. The object is to read in an image from an external file and divide this image into small tile sections. The program requires two inputs.

First, there is the input, which was already used with the 100-point version for the grid size.

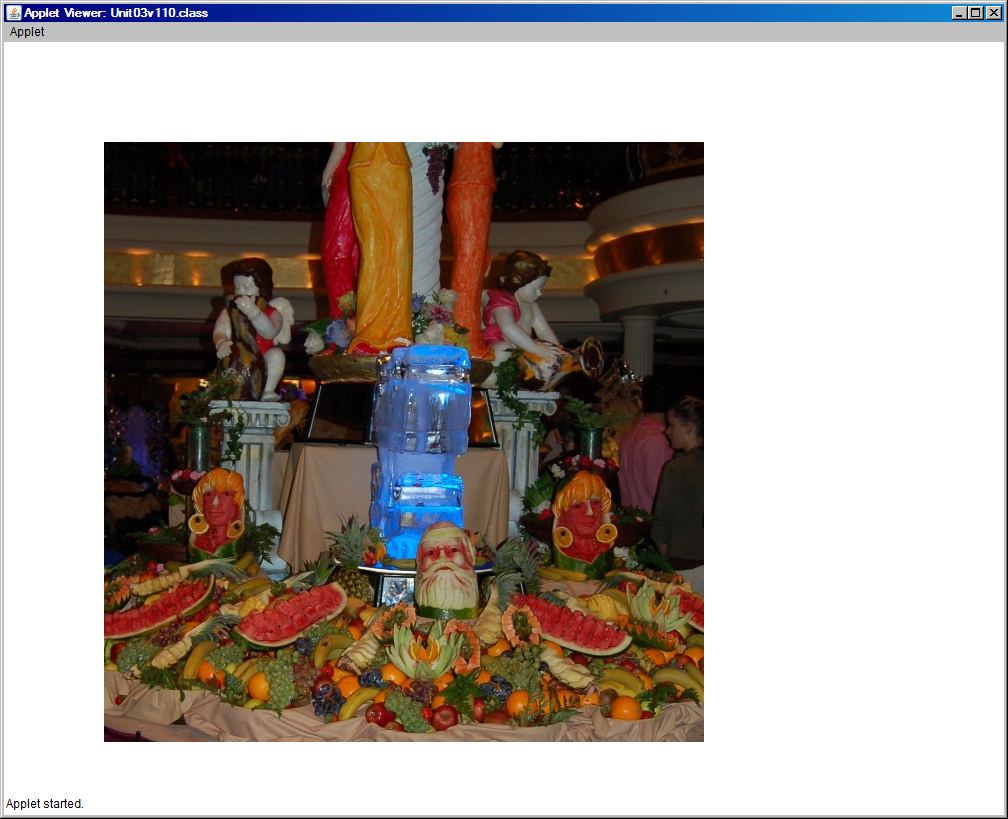
Second, there is the input for the image file name to be used by the program.

The two inputs are shown below.





At the start of the program the original **Food.JPG** image file must be displayed for five seconds as shown below. That is not how the image will look when the puzzle starts. Your program needs to divide the original file into sub images of small square tiles.



The tiles in the image needs to be randomly arranged, just like the letters in the Letter Puzzle. The execution below is one example of the start of the image puzzle for a 5 X 5 grid.

