CROSSWORD

Cluster Innovation Centre

University of Delhi

ANSHIKA – 11909
BHAVYA VERMA – 11914

April 2020

Month Long Project submitted for the paper

Reflecting thought processes through Object Oriented Programming

Certificate of Originality

The work embodied in this report entitled **"Crossword"** has been carried out by **Anshika** and **Bhavya Verma** for the paper "**Reflecting thought processes through Object Oriented Programming"**. We declare that the work and language included in this project report is free from any kind of plagiarism.

ANSHIKA
BHAVYA VERMA

Abstract

Reflecting thought processes through Object Oriented Programming

by

Anshika, Bhavya Verma

Cluster Innovation Centre, 2020

This project reports on the construction of a Crossword Puzzle, designed with the help of Java Programming Language. The project incorporates the use of application of various concepts of Object-Oriented Programming which has helped us in depicting our project on a Virtual Platform, while making use of Graphical User Interface (GUI).

The puzzle consists of words, both across and down as in the case of an original Crossword Puzzle.

Our puzzle consists of words corresponding to number of clues or definitions that are supplied into correspondingly fitted number of squares, i.e. one letter per square, the words being arranged horizontally and vertically so that most of the letters form two parts of a word.

The puzzle consists of a Solved Clue Support which can be enabled, if the player wishes to know whether his guess is correct or not. This Support provides the word details that has been answered correctly by the player.

Once the player has filled all the words correctly and all such word details appear in the solved clue support space the crossword puzzle is complete and the game ends.

INTRODUCTION

With the help of different Object-Oriented Programming concepts, we are able to construct different parts and functioning of the different units of the Crossword Project. Most extensively used concept is the use of **Classes**, since all the work in Java is done with the help of the Classes and creating their **Objects** and using them that helps in a smooth working of the project.

The private methods and variables defined in the classes, hides them from the other classes and can be accessed only through the member functions of the class they're defined in, implementing **Encapsulation**.

Almost all the Classes that are being used, make use of Constructors, which is included in **Polymorphism**, since Constructors have the same name as of the Classes, to initialize the variables contained in them. Moreover, in this project, we make use of overriding various functions, which is considered a type of **Polymorphism**.

Since in this project, we are working with graphics, we need to include various pre-existing classes in the beginning of the file and the classes we make inherit them using the keyword "extends", making the use of **Inheritance**.

Also overriding the methods of Abstract Superclass is a concept of Object-Oriented Programming called as **Abstraction.**

The Crossword project consists of 11 java files, containing various components of the Crossword.

1. **Cell.java**

   All the cells in the Crossword Grid are Classified into two types, Solid Cells, the cells that are not a part of any clue and Character Cells, which are a part of the clue.

   The file Cell.java contains the Abstract class to allow both Solid Cells and Character Cells to be added to the same arrays.

2. **SolidCell.java**

   Fills the background black for all the cells that are not a part of any clue

3. **ClueSolver.java**

   Forms the string representation of the Statement to be printed on the Clue Solver, which includes the name of the person solving the crossword and the date and time on which the clue was guessed correctly.

4. **Clue.java**

   Contains a Class that represents a clue and its answer in the crossword.

5. **CharacterCell.java**

   Contains a Class for the characteristics of all the cells that form a part of the Clue.

6. **Crossword.java**

   This file represents the Crossword that is currently being played, setting all the Clues, setting the Username etc.

7. **GridPanel.java**

   Grid Panel represents the crossword on the Screen. Involves creating Cells, drawing the necessary components and adding them to the Grid.

8. **SolvedLog.java**

   Contains a Class that displays the Log of the Solved Clues on the Solved Clues Panel, in a chronological order.

9. **CrosswordIO.java**

   This file handles all the input/output operations of the Crossword Puzzle, including reading from a file, writing onto a file.

10. **CrosswordFrame.java**

    Contains the Main Function and all the Swing API components for the Layout of the Crossword Puzzle. The described layout is included in this file.

11. **CrosswordExample.java**

    Contains the example Crossword that helps us to test the correct functionality of the Project.

METHODOLOGY

In the construction of the Crossword, we have some major components that contribute in the formation of the layout. We have the Crossword Grid, Name panel, Load and Save Buttons, Clue Panel and the Clue Solver Panel. Each file of the project incorporates the properties and functions of these components.

The most basic component in the Crossword grid are the Cells. In the making of the Crossword, we have classified Cells into two types, **Solid Cells**, the cells which are not a part of the clue and **Character Cells**, which form a part of the Clue**.**

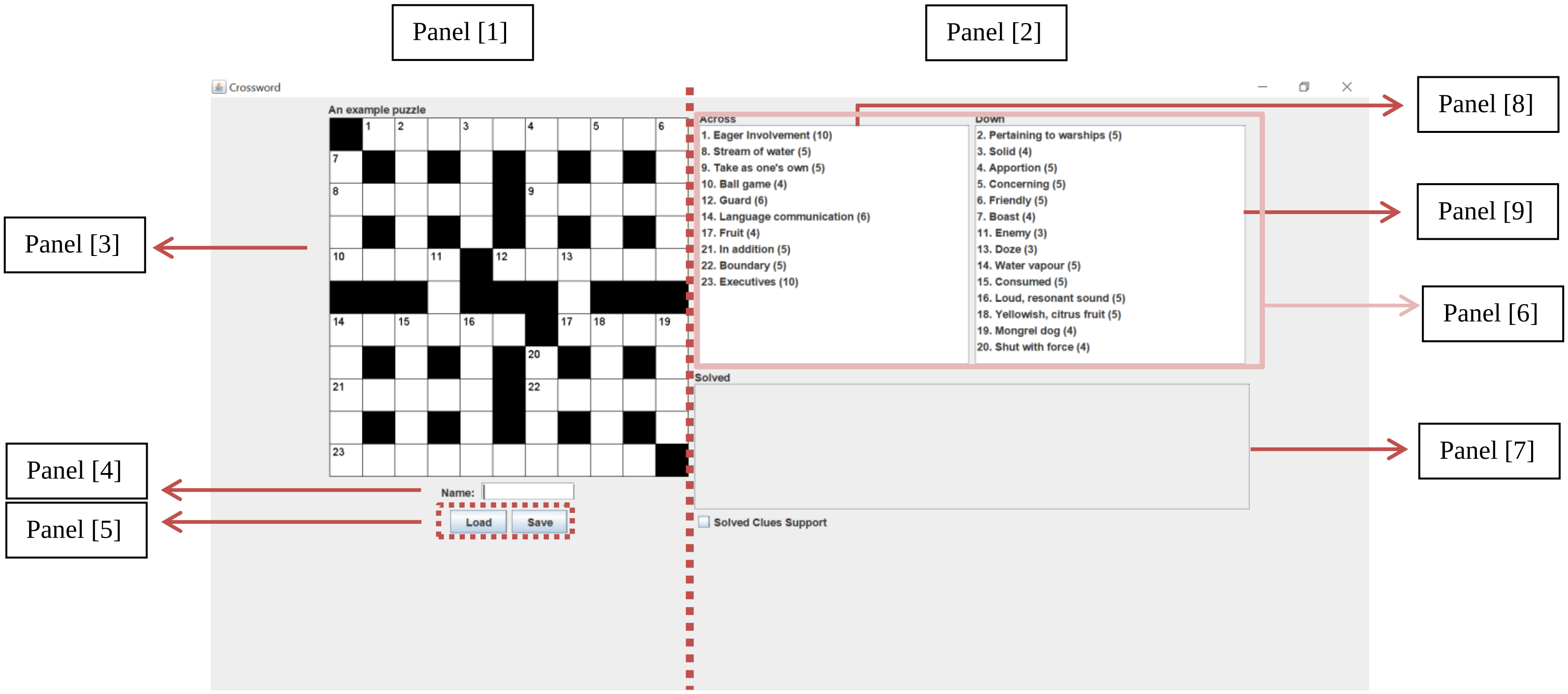
The layout for the Crossword on the Virtual Screen is made with the help of Panels, included in the Swing API (Application Programming Interface), i.e. a set of extensible GUI Components for Java based GUI Applications.

The Layout is made with the help of Panels in the form of a tree, the base Frame is divided into 2 Panels, Left (panel [1]) and the Right (panel [2]), analogous to the Left and the Right Child of a Tree.

In the Left Panel (panel [1]), adding the Crossword Grid towards the north (panel [3]), the Name Label in the center (panel [4]), the Load and Save Buttons in the South (panel [5]).

In the Right Panel (panel [2]), adding a Panel [6] in the north, which is further divided into two panels, Panel [8] on the Left, which accounts for the Across Clues and Panel [9] on the Right which accounts for the Down Clues. We also have the Solved Panel (panel [7]) in the South, which contains a Solved Clues

Checkbox when clicked, displays the Solved Log Panel that provides the details of the correctly answered clue, including the date and time as well as the name of the person currently playing the game.



Across

1. Eager Involvement (10)
8. Stream of water (5)
9. Take as one's own (5)
10. Ball game (4)
12. Guard (6)
14. Language communication (6)
17. Fruit (4)
21. In addition (5)
22. Boundary (5)
23. Executives (10)

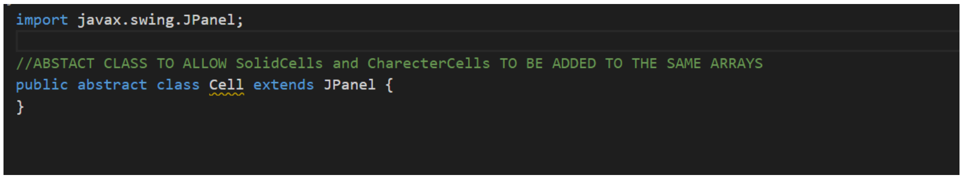
Down

2. Pertaining to warships (5)
3. Solid (4)
4. Apportion (5)
5. Concerning (5)
6. Friendly (5)
7. Boast (4)
11. Enemy (3)
13. Doze (3)
14. Water vapour (5)
15. Consumed (5)
16. Loud, resonant sound (5)
18. Yellowish, citrus fruit (5)
19. Mongrel dog (4)
20. Shut with force (4)

Describing each File included in detail:

1. **Cell.java**

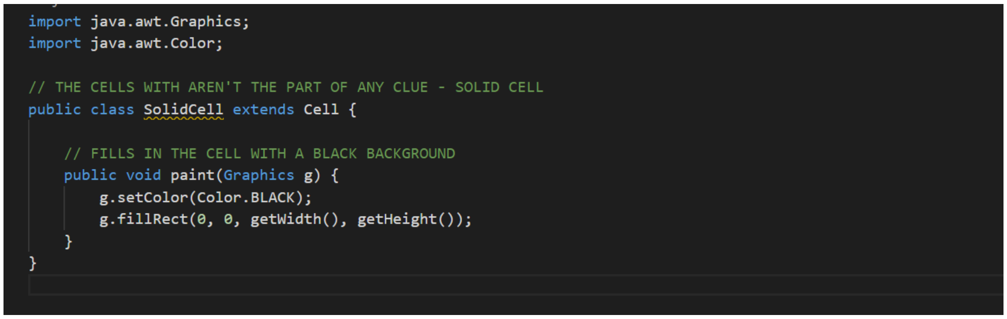
   The file includes JPanel, which is a part of Java Swing package that can store a group of components. The file includes an abstract class "Cell" that extends JPanel, such that the Solid Cells and the Character Cells can be added to the same arrays.



```java
import javax.swing.JPanel;

//ABSTACT CLASS TO ALLOW SolidCells and CharecterCells TO BE ADDED TO THE SAME ARRAYS
public abstract class Cell extends JPanel {
}
```

2. **SolidCell.java**

   Includes a class "SolidCell" that inherits the super class "Cell", that paints all the cells that are not a part of the Clue as BLACK.

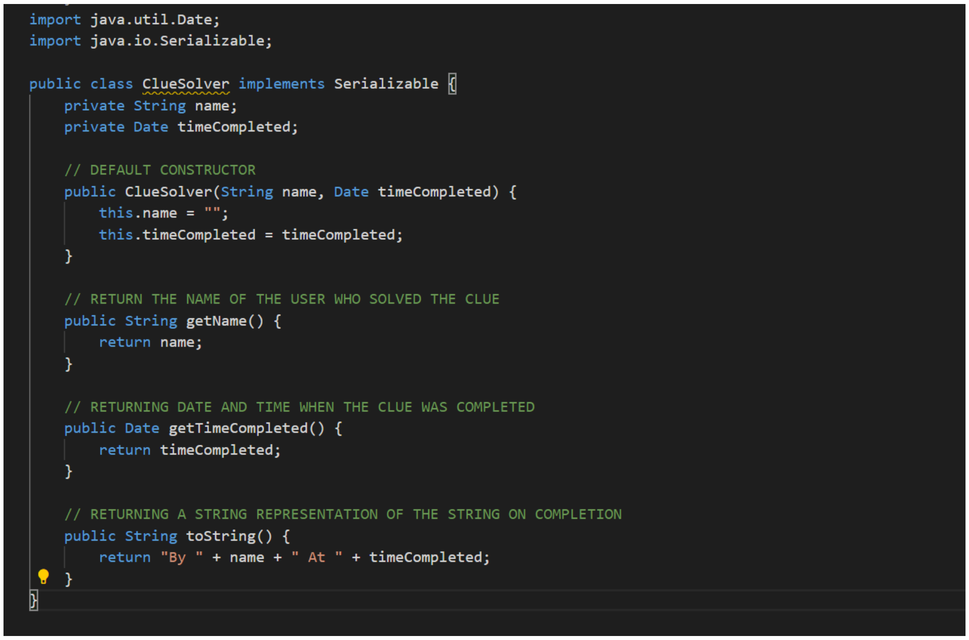


```java
import java.awt.Graphics;
import java.awt.Color;

// THE CELLS WITH AREN'T THE PART OF ANY CLUE - SOLID CELL
public class SolidCell extends Cell {

    // FILLS IN THE CELL WITH A BLACK BACKGROUND
    public void paint(Graphics g) {
        g.setColor(Color.BLACK);
        g.fillRect(0, 0, getWidth(), getHeight());
    }
}
```

3. **ClueSolver.java**

   The file contains a Class "ClueSolver" that sets the Name of the person currently playing the game and the date and time when the clue was guessed correctly, with the help of a Constructor.

Also contains respective functions that return each variable and another function that returns the string representation on the time of completion of a particular clue.



```
import java.util.Date;
import java.io.Serializable;

public class ClueSolver implements Serializable {
    private String name;
    private Date timeCompleted;

    // DEFAULT CONSTRUCTOR
    public ClueSolver(String name, Date timeCompleted) {
        this.name = "";
        this.timeCompleted = timeCompleted;
    }

    // RETURN THE NAME OF THE USER WHO SOLVED THE CLUE
    public String getName() {
        return name;
    }

    // RETURNING DATE AND TIME WHEN THE CLUE WAS COMPLETED
    public Date getTimeCompleted() {
        return timeCompleted;
    }

    // RETURNING A STRING REPRESENTATION OF THE STRING ON COMPLETION
    public String toString() {
        return "By " + name + " At " + timeCompleted;
    }
}
```

4. Clue.java

Contains a class "Clue" that represents a clue and its answer in the crossword, with members including,

number – The number of the clue

x and y – The respective x and y coordinates of the first letter of the clue

clue – The hint given to the user

answer – The solution to the clue

Also contains a function that returns true if the answer is correct, i.e. the clue matches the answer, another function that sets the position of the cell for the answer entered by the user, and if the clue is answered correctly, it also updates the Solved Log Panel. A function to get the character entered by the user at the designated position.

The class also contains a special function that produces a bracketed string of the length of the answer, which takes into account the answers containing spaces and hyphens.

5. **CharacterCell.java**

The class "CharacterCell" that extends the super class "Cell", represents the cell that contains a character of the clue.

The class calls its constructor which invokes the constructor of its super class, and initializes all variables, allows the cell to set focus that helps the user to know that it will be the field that receives the keyboard input and using the "tab" button the user can jump to the next field.

Also, the Class methods uses various listeners such as MouseListener, KeyListener and FocusListener, Listener is a user interface listener, a method which is called when the user does something (e.g. click a button) that the programmer has indicated they want to be notified of. The listener method is passed an event parameter that may tell something about what happened.

The class contains methods so as to associate Across and Down clues with the cell and calculate the default orientation of the Clue, i.e. Across or Down, moreover when the Character cells are focused, they turn Yellow and are added a Red border around them, to let the player easily distinguish between the regular and the focused set of cells.

6. **Crossword.java**

The file contains a class “Crossword” that represents the Crossword Puzzle that the user is currently playing. The class implements “Serializable” interface, where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object.

The default constructor initializes the variables i.e.

title – The title of the Crossword

size – The size of the Crossword Grid

acrossClues – The clues that represent Across

downClues – The clues that represent Down

Function to check whether the clue fits on the grid, if not then tries to solve the problem by resizing the grid.

Creating and returning a list of clues combined into a list.

7. **GridPanel.java**

   Class “GridPanel” represents the Crossword on the screen, that inherits the properties of “JPanel”. The constructor initializes the variables included,

   width – The width of the onscreen component

   height – The height of the onscreen component

   crossword – The Crossword associated with the panel

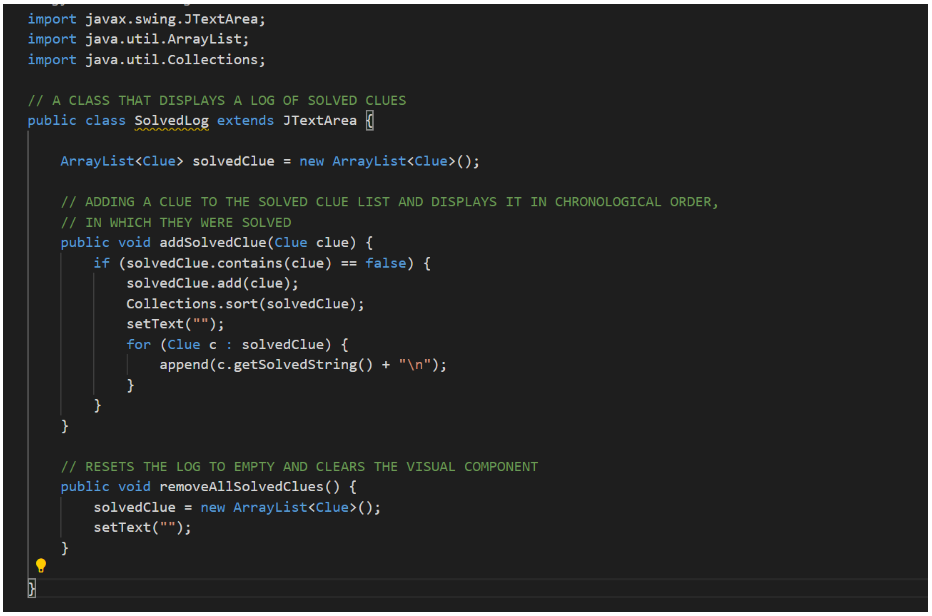
   The constructor also sets up the Grid, creating cells and panels, initially fills all the cells with solid cells and then represent those as character cells which form a part of a clue.

   The class also contains functions to highlight certain cells when focused and draws components associated with Grid Panel on the screen.

8. **SolvedLog.java**

   Contains a class “SolvedLog” that displays a log of solved clues, inherits the properties of “JTextArea” which represents a multi-line area that displays text. It is used to edit the text.

   The class contains functions that adds a clue to the solved clue list and displays it on the screen in chronological order, in which they were solved, also resets the log empty and clears the visual component.



```java
import javax.swing.JTextArea;
import java.util.ArrayList;
import java.util.Collections;

// A CLASS THAT DISPLAYS A LOG OF SOLVED CLUES
public class SolvedLog extends JTextArea {

    ArrayList<Clue> solvedClue = new ArrayList<Clue>();

    // ADDING A CLUE TO THE SOLVED CLUE LIST AND DISPLAYS IT IN CHRONOLOGICAL ORDER,
    // IN WHICH THEY WERE SOLVED
    public void addSolvedClue(Clue clue) {
        if (solvedClue.contains(clue) == false) {
            solvedClue.add(clue);
            Collections.sort(solvedClue);
            setText("");
            for (Clue c : solvedClue) {
                append(c.getSolvedString() + "\n");
            }
        }
    }

    // RESETS THE LOG TO EMPTY AND CLEARS THE VISUAL COMPONENT
    public void removeAllSolvedClues() {
        solvedClue = new ArrayList<Clue>();
        setText("");
    }

}
```

9. CrosswordIO.java

The file contains class “CrosswordIO” that contains functions that handles all of the input output operations of the crossword using input output stream packages, the function of the load and save button also making the use of exception handling to print errors.

10. CrosswordFrame.java

The file contains the class “CrosswordFrame” includes the main method, which is called when the program starts running.

The class also includes a variety of Swing Components, creates the various components like labels and lists, initializes the grid, adds all the clues to the list and ensures that the solved log is empty.

This class is responsible for the layout of all the components of the Crossword, creating mouse listeners, keeping the solved clues in a scroll panel.

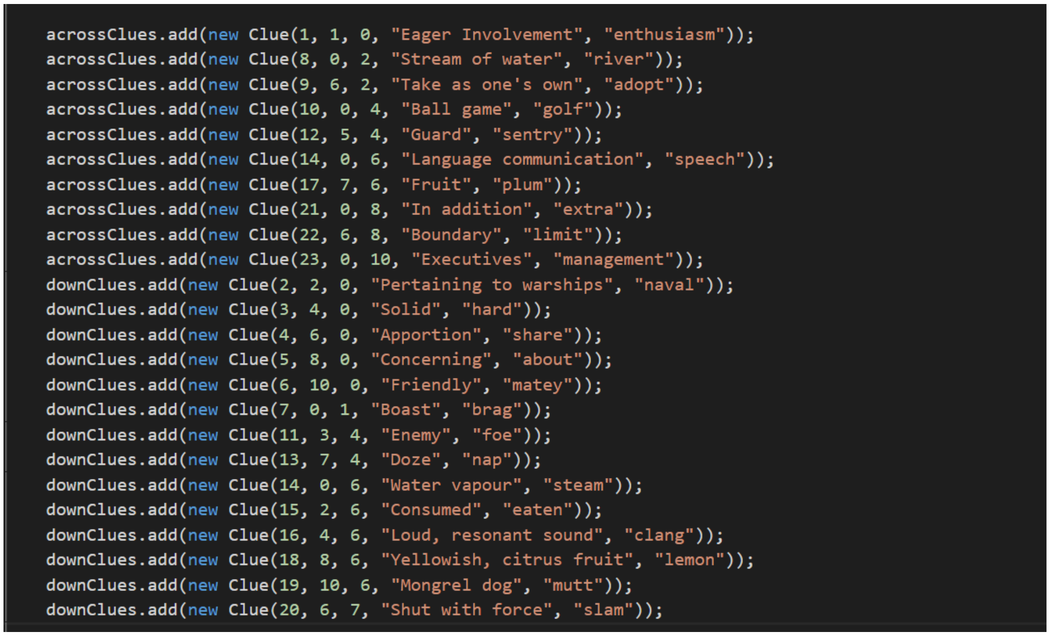
The Layout is made with the help of Panels in the form of a tree, the base Frame is divided into 2 Panels, Left (panel [1]) and the Right (panel [2]), analogous to the Left and the Right Child of a Tree.

In the Left Panel (panel [1]), adding the Crossword Grid towards the north (panel [3]), the Name Label in the center (panel [4]), the Load and Save Buttons in the South (panel [5]).

In the Right Panel (panel [2]), adding a Panel [6] in the north, which is further divided into two panels, Panel [8] on the Left, which accounts for the Across Clues and Panel [9] on the Right which accounts for the Down Clues. We also have the Solved Panel (panel [7]) in the South, which contains a Solved Clues Checkbox when clicked, displays the Solved Log Panel that provides the details of the correctly answered clue, as described above in the file.

11. **CrosswordExample.java**

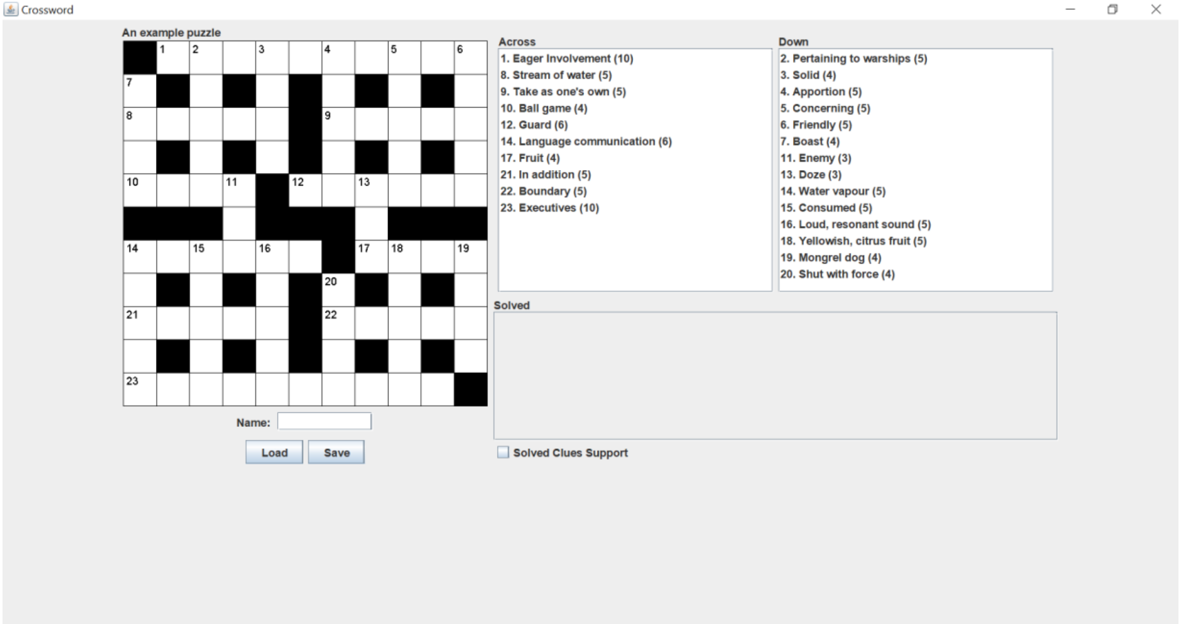
The file contains class "CrosswordExample", which sets up an example crossword that we use to test the correct functionality of our program.



```
acrossClues.add(new Clue(1, 1, 0, "Eager Involvement", "enthusiasm"));
acrossClues.add(new Clue(8, 0, 2, "Stream of water", "river"));
acrossClues.add(new Clue(9, 6, 2, "Take as one's own", "adopt"));
acrossClues.add(new Clue(10, 0, 4, "Ball game", "golf"));
acrossClues.add(new Clue(12, 5, 4, "Guard", "sentry"));
acrossClues.add(new Clue(14, 0, 6, "Language communication", "speech"));
acrossClues.add(new Clue(17, 7, 6, "Fruit", "plum"));
acrossClues.add(new Clue(21, 0, 8, "In addition", "extra"));
acrossClues.add(new Clue(22, 6, 8, "Boundary", "limit"));
acrossClues.add(new Clue(23, 0, 10, "Executives", "management"));
downClues.add(new Clue(2, 2, 0, "Pertaining to warships", "naval"));
downClues.add(new Clue(3, 4, 0, "Solid", "hard"));
downClues.add(new Clue(4, 6, 0, "Apportion", "share"));
downClues.add(new Clue(5, 8, 0, "Concerning", "about"));
downClues.add(new Clue(6, 10, 0, "Friendly", "matey"));
downClues.add(new Clue(7, 0, 1, "Boast", "brag"));
downClues.add(new Clue(11, 3, 4, "Enemy", "foe"));
downClues.add(new Clue(13, 7, 4, "Doze", "nap"));
downClues.add(new Clue(14, 0, 6, "Water vapour", "steam"));
downClues.add(new Clue(15, 2, 6, "Consumed", "eaten"));
downClues.add(new Clue(16, 4, 6, "Loud, resonant sound", "clang"));
downClues.add(new Clue(18, 8, 6, "Yellowish, citrus fruit", "lemon"));
downClues.add(new Clue(19, 10, 6, "Mongrel dog", "mutt"));
downClues.add(new Clue(20, 6, 7, "Shut with force", "slam"));
```

RESULT

Using various Object-Oriented Programming concepts and pre-defined classes and packages, we were able to construct a Crossword puzzle that is very similar to the real-life Crossword puzzles. With the help of various Swing API components, we were able to provide the required layout for the Crossword items like the Grid, the across and down clues, and the Solved Clues Support.



Across

1. Eager Involvement (10)
8. Stream of water (5)
9. Take as one's own (5)
10. Ball game (4)
12. Guard (6)
14. Language communication (6)
17. Fruit (4)
21. In addition (5)
22. Boundary (5)
23. Executives (10)

Down

2. Pertaining to warships (5)
3. Solid (4)
4. Apportion (5)
5. Concerning (5)
6. Friendly (5)
7. Boast (4)
11. Enemy (3)
13. Doze (3)
14. Water vapour (5)
15. Consumed (5)
16. Loud, resonant sound (5)
18. Yellowish, citrus fruit (5)
19. Mongrel dog (4)
20. Shut with force (4)