This is a program intended to be a randomly generated crossword. Users can specify the dimensions and topic of the crossword. The program will generate a square crossword with no black spaces as a Java Graphical object. This is intended to be a tool for caretakers of small children as an easy way to generate activities based on learning vocabulary.

Classes:

Crossword

1. Semantics and Use
    a. This is our Main class from which everything executes. There is only one instance of this per execution. This takes in the keyboard input and either prints out a crossword if valid or shows an error if not. It also takes in arguments from the command line, parses them for errors and calls the constructors of our crossword generation and graphics classes.
2. Member variables
    a. None

3. Constructor
    a. This was the main class so there is no constuctor
4. Methods
    a. ShowUsageandExit()
        i. Quits program if the input is not correctly used, while printing out the correct input format
        ii. This is public as it is called from the main method
        iii. This is non-recursive

Cell

1. Semantics and Use
    a. This is our Cell class that stores each letter for the crossword. There are several instances of this class within each instance of GeneratedGrid()
2. Member variables
    a. Private string Output
        i. This variable holds the letter for the specified cell. This is private as we do not want it to be accessed outside this class without the helper methods we created. It is a string as this is a crossword puzzle.

3. Constructor
    a. this.output = "";
        i. This sets the default value of the Output string as a blank. This is basically a placeholder value until an actual string is set as the output.
4. Methods
    a. public String getOutput()
        i. This is a helper method to get the output of the cell
        ii. It is public as it must be called outside the class
        iii. It is non-recursive
    b. public void setOutput(String word)
        i. Helper method to set output of the cell
        ii. Public as it must be called outside the class
        iii. Non-recursive

Visual (subclass of JPanel)

1. Semantics and Use
    a. This is the class used to generate the printout of the Crossword
2. Member Variables
    a. GeneratedGrid genGrid;
        i. This is the grid to be printed
    b. private static final int size = 75;
        i. This is the preset size we decided for the size of the box
        ii. This is private as it should not be accessed outside the class

3. Constructor
    a. genGrid = gGrid;
        i. This sets the genGrid equal to the inputted GeneratedGrid
    b. int dimension = GeneratedGrid.getDimension();
        i. This sets the dimension from the inputted GeneratedGrid
    c. setLayout(new GridLayout(dimension, dimension));
        i. This sets the layout of the Visual object as a grid
    d. this.setPreferredSize(new Dimension(dimension * size, dimension * size));
        i. This sets the size of the Visual object
    e. for (int i = 0; i < dimension; i++) { for (int j = 0; j < dimension; j++) {String word = genGrid.grid[i][j].getOutput(); this.add(new Letter(word)}}
        i. This for loop generates a Letter object for each Cell in the GeneratedGrid and adds it into the Visual object

4. Methods
    a. public void paintComponent(Graphics g)
        i. This is the overriden paintComponent method
        ii. This is public just like the original paintComponent method
    b. public void display()
        i. This is the method that actually sets the JFrame to be visible
        ii. This must be called in other classes so it must be public

Letter (JButton Subclass)

1. Semantics and Use
    a. This creates a JButton subclass with a square form and the character from the corresponding Cell
2. Member Variables
    a. There are no unique Member Variables apart from the ones in the JButton class

3. Constructor
    a. There is one constructor that properly sets the graphical details we wanted
        i. input = input.toUpperCase();
        ii. setText(input);
        iii. this.setFont(new Font("Arial", Font.PLAIN, 40));
        iv. this.setOpaque(true);
        v. this.setBorderPainted(true);
        vi. }
4. Methods
    a. There are no specific methods for the Letter Class

GeneratedGrid

1. Semantics and Use
2. Member Variables
    a. public static ArrayList<String> wordbank
        i. This is the ArrayList that contains all the words we are drawing the crossword from
        ii. It is public as it can be accessed from methods
    b. private static int N
        i. This is the dimension of the generatedGrid
        ii. Private as it should only be accessed through helper methods
    c. public Cell[][] grid

    i. The Cell array that will be used to house the letters
    ii. Public as it must be accessed from the Visual class
  d. int midPoint
    i. Int for the midpoint of the GeneratedGrid. Used in the brute force method generation
  e. Visual visual
    i. Instance of the visual object to be used to print out the crossword
  f. static String lang
    i. String to be used as a flag for deciding the type of wordbank
3. Constructor
  a. public GeneratedGrid(int dimension, String language) {
  b. N = dimension;
    i. This sets the dimension from the keyboard input for the GeneratedGrid
  c. lang = language;
    i. This sets the lang flag equal to the input from the keyboard
  d. if (lang.equals("standard")) { wordbank = wordScanner("wordlist.txt", N);}else if (lang.equals("names")) {wordbank = wordScanner("names.txt", N);} else if (lang.equals("acronyms")) {wordbank = wordScanner("acronyms.txt", N);} else {Crossword.showUsageAndExit();}
    i. This sets the wordbank based on the lang flag

  e. this.grid = new Cell[N][N];
    i. This initializes the cell grid
  f. this.initializeCells(this.grid);
    i. Initializes the cells in the grid
  g. this.midPoint = (int) getDimension() / 2;
    i. Sets midpoint based on dimension
4. Methods
  a. public void wordInputWithInit()
    i. Brute force method for inputting words into generated grid

  b. public boolean checkInput()
    i. Checks if there are possible inputs given the current state of the grid
  c. public boolean checkGrid()
    i. verifies that for every possible down or across, there is at least one possible word
  d. void insertWordVert(String word, int index)
    i. Inserts a word into the grid vertically
  e. void insertWordHoriz(String word, int index)

    i. Inserts a word into the grid vertically
f. public void removeWord(int index)
    i. Removes a word
g. private void clearGrid()
    i. Wipes the grid completely
h. private void removeWordUnder(int index)
    i. Removes the word under the top row. This prevents the remove word method from disrupting the initial grid.
i. private void clearGridWithInput()
    i. Specifically clears the entire grid except for the intial input
j. private void initializeCells(Cell[][] grid)
k. public boolean isComplete()
    i. Checks if every Cell in the grid has some input
l. public boolean isCorrect()
    i. Checks if crossword is valid
m. public ArrayList<String> possibleWords(int index, ArrayList<String> wordbank)
    i. Creates an ArrayList of possible words that can be placed at the row position corresponding to the index given
n. private String regex(String word)
    i. Creates the regex for a word
o. private void createInitialGrid(ArrayList<String> wordbank)
    i. creates the initial grid for the wordinput method
p. public ArrayList<String> possibleWordsVert(int index, ArrayList<String> wordbank)
    i. Creates an ArrayList of possible words that can be placed at the column position corresponding to the index given
q. public void printPossibles()
    i. Prints all possible wordbank. Used for testing
r. public String getCell(int x, int y)
    i. Gets the input in a specific Cell in the grid
s. public static String chooseWord(ArrayList<String> wordbank)
    i. Chooses a word randomly from wordbank
t. public boolean matches(GeneratedGrid someGrid)
    i. Used to see if a grid matches another grid. the standard Java .equals(Object o) method was insufficient.

u. private static ArrayList<String> wordScanner(String filename, int N)
    i. Scans the initial word bank text file.
v. public void addWord(int n)

i. Chooses a word from the i-th most vulnerable across or down. Most vulnerable means the smallest possible word bank. (For depthsearch)

w. public ArrayList<String> getWordBank()
   i. Getter method for the wordbank

x. public static int getDimension()
   i. Getter method for the Dimension

y. public static String getLang()
   i. Getter method for the wordbank flag

z. private static boolean isClean(String string)
   i. Determines if a string contains only ASCII characters

Node (subclass of generatedGrid)

1. Semantics and Use
   a. This is the node class for the DepthFirstSearch linked list.
2. Member Variables
   a. GeneratedGrid grid
      i. This is the GeneratedGrid within the node
   b. boolean checkGrid;
      i. This is the boolean for whether the grid has possible words or not
   c. boolean isCorrect
      i. This boolean checks if the node is a valid crossword
3. Constructor
   a. Constructor 1
      i. this.grid = grid;
      ii. isCorrect = grid.isCorrect();
      iii. checkGrid = grid.checkGrid();
      iv. Used for the initial starting node
   b. Constructor 2
      i. this.isCorrect = node.isCorrect
      ii. int n = GeneratedGrid.getDimension()
      iii. GeneratedGrid clonedGrid = new GeneratedGrid(n, GeneratedGrid.getLang())
      iv. for (int i = 0; i < n; i++) { for (int j = 0; j < n; j++) { clonedGrid.grid[i][j].setOutput(node.grid.grid[i][j].getOutput()); }}
      v. this.grid = clonedGrid;
      vi. this.checkGrid = clonedGrid.checkGrid();
      vii. Used for generating consequent nodes based on previous node. This was necessary because when a new Node created from a previous Node results in both nodes referencing the same grid field in memory. Thus, it was

easier to copy the contents of each cell into a new cell[][] in the copied node.

4. Methods
    a. public static boolean contains(ArrayList<Node> nodes, Node node)
        i. This is to check whether the Node in question is in the Arraylist of Nodes

DepthFirstSearch (subclass of GeneratedGrid)

1. Semantics and Use
    a. This is used for the Depth First Search crossword generation algorithm. It iteratively implements the DFS algorithm using a stack.
2. Member Variables
    a. Node startNode;
3. Constructor
    a. super(getDimension(), GeneratedGrid.getLang());
    b. this.startNode = start;
4. Methods
    a. public GeneratedGrid DFS()
        i. This method takes in a startGrid (always an empty grid). It uses the addWord() method in the GeneratedGrid class to create the possible children of every node. When it creates a complete and correct crossword grid, it returns a GeneratedGrid object.

PrintGrid

1. Semantics and Use
    a. This is the class used to house the method used for the in console printing of the GeneratedGrid
2. Member Variables
    a. No member variables
3. Constructor
    a. There is no constructor as there are no necessary member variables
4. Methods
    a. public static void noGraphicsPrint(GeneratedGrid genGrid)
        i. This is the method for in console printing of the GeneratedGrid