Button

```
package com.SpotTheImage;

import javax.swing.*;
import java.awt.*;

public class Button extends JButton {

        private static final long serialVersionUID = 1L;
        private String text;
    private int x;
    private int y;
    private int width;
    private int height;

    public Button(String text, int x, int y, int width, int height) {
        this.text = text;
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;

        this.setBounds(this.x, this.y, this.width, this.height);
        this.setText(this.text);
        this.setFocusable(false);
    }

    public Button(String text) {
        this.text = text;

        this.setText(this.text);
        this.setFocusable(false);
    }
}

class PlayButton extends Button {

    private JPanel container;
    private CardLayout cardLayout;

    public PlayButton(String text, int x, int y, int width, int height, JPanel container, CardLayout
cardLayout) {
        super(text, x, y, width, height);
        this.container = container;
        this.cardLayout = cardLayout;
    }


    public void swapCard(String cardNum) {
        this.cardLayout.show(this.container, cardNum);
    }
}

class ExitButton extends Button {

    public ExitButton(String text, int x, int y, int width, int height) {
        super(text, x, y, width, height);
    }

}

class BackButton extends Button {

    private JPanel container;
    private CardLayout cardLayout;

    public BackButton(String text, int x, int y, int width, int height, JPanel container, CardLayout
cardLayout) {
        super(text, x, y, width, height);
        this.container = container;
        this.cardLayout = cardLayout;
    }

    public BackButton(String text, JPanel container, CardLayout cardLayout) {
        super(text);
        this.container = container;
        this.cardLayout = cardLayout;
    }

    public void swapCard(String cardNum) {
        this.cardLayout.show(this.container, cardNum);
    }
}

class CategoryButton extends Button {
    private String category;
    private JPanel container;
    private CardLayout cardLayout;

    public CategoryButton(String text, JPanel container, CardLayout cardLayout) {
        super(text);
        this.container = container;
        this.cardLayout = cardLayout;
    }


    public void swapCard(Words words, MainGamePanel mainGamePanel) {

        Word word = words.selectRandomWord();
        word.splitWordToLetters();
        mainGamePanel.setRandomWord(word);
        mainGamePanel.setGuessedLetters(word);
        this.cardLayout.show(this.container, "3");
    }

    public String getCategory() {
        return this.category;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}

class NewButton extends Button {


            public NewButton(String text, int x, int y, int width, int height) {
        super(text, x, y, width, height);
    }
}
```

Homescreen

```
package com.SpotTheImage;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Homescreen extends JPanel implements ActionListener {

        private static final long serialVersionUID = 1L;
        final String[] categoryButtonsText = {"Play"};
    private CategoryButton[] categoryButtons = new CategoryButton[3];
    private BackButton backButton;
    private MainGamePanel mainGamePanel;
    private Words words;

    public Homescreen(int WIDTH, int HEIGHT, ImageIcon backgroundImg, JPanel container,
CardLayout cardLayout, MainGamePanel mainGamePanel, Words words) {
        this.mainGamePanel = mainGamePanel;
        this.words = words;

        JLabel categoryScreenBg = new JLabel(backgroundImg);

        this.setLayout(new BorderLayout());
        this.add(categoryScreenBg);

        categoryScreenBg.setLayout(null);

        JPanel buttonsPanel = new JPanel();
        int buttonsPanelWidth = 500;
        int buttonsPanelHeight = 50;
        int buttonsPanelX = (WIDTH / 2) - (buttonsPanelWidth / 2);
        int buttonsPanelY = HEIGHT - (buttonsPanelHeight * 3);
        buttonsPanel.setBounds(buttonsPanelX, buttonsPanelY, buttonsPanelWidth,
buttonsPanelHeight);
        buttonsPanel.setLayout(new GridLayout());

        categoryScreenBg.add(buttonsPanel);

        for (int i=0; i<this.categoryButtonsText.length; i++) {
            CategoryButton categoryButton = new CategoryButton(this.categoryButtonsText[i],
container, cardLayout);
            categoryButton.setCategory(this.categoryButtonsText[i].toLowerCase());
            this.categoryButtons[i] = categoryButton;
            categoryButton.addActionListener(this);
            buttonsPanel.add(categoryButton);
        }
        String backButtonText = "Exit";
        this.backButton = new BackButton(backButtonText, container, cardLayout);
        this.backButton.addActionListener(this);
        buttonsPanel.add(this.backButton);


    }

    @Override
    public void actionPerformed(ActionEvent e) {


         if (e.getSource() == this.backButton) {
             System.exit(0);
         }
    }
}
```

Keyboard

```
package com.SpotTheImage;


import javax.swing.*;

public class Keyboard extends JButton {

            //our virtual keyboard
            private static final long serialVersionUID = 1L;
            private char[] letters = {
            'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p',
            'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l',
            'z', 'x', 'c', 'v', 'b', 'n', 'm'
    };
    private QwertyButton[] buttons = new QwertyButton[26];

    public Keyboard() {

        for (int i=0; i<this.letters.length; i++) {
            this.buttons[i] = new QwertyButton(this.letters[i]);
        }
    }
    public void displayButtons(JPanel keyboardPanel) {

        for (QwertyButton button : this.buttons) {
            keyboardPanel.add(button);
        }
    }

    public void resetKeyboard() {

        for (QwertyButton button : this.buttons) {
            button.setEnabled(true);
            button.setBackground(null);
        }
    }

    public void disableKeyboard() {

        for (int i=0; i<this.buttons.length; i++) {
            this.buttons[i].setEnabled(false);
        }
    }

    public QwertyButton[] getButtons() {

        return this.buttons;
    }
}
```

Main

```
package com.SpotTheImage;
import javax.swing.*;
import java.awt.*;

public class Main {

    public static void main(String[] args) {

        final int WIDTH = 1200;
        final int HEIGHT = 800;
        final String TITLE = "SpotTheImage (Word Game)";
        // the file which contains our home screen image
        final ImageIcon backgroundImg = new
ImageIcon("C:/Users/Eyad/eclipse-workspace/CSC-330/src/com/SpotTheImage/HS2.png");



        JFrame screen = new JFrame();
        CardLayout cardLayout = new CardLayout();
        screen.setSize(new Dimension(WIDTH, HEIGHT));
        screen.setTitle(TITLE);
        screen.setLocationRelativeTo(null);
        screen.setResizable(false);
        screen.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        JPanel container = new JPanel();
        container.setLayout(cardLayout);
        screen.add(container);

        Words words = new Words();


        MainGamePanel mainGamePanel = new MainGamePanel(WIDTH, HEIGHT, container,
cardLayout, words);
        Homescreen homescreen = new Homescreen(WIDTH, HEIGHT, backgroundImg,
container, cardLayout, mainGamePanel, words);

   // we have the option to select which screen gets appeared first, when the game starts
        container.add(homescreen, "2");
        container.add(mainGamePanel, "3");

        screen.setVisible(true);
    }
}
```

MainGamepanel

```
package com.SpotTheImage;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MainGamePanel extends JPanel implements ActionListener {
   /**
         *
         */
        private static final long serialVersionUID = 1L;
        private int WIDTH;
   private int HEIGHT;
   private Image backgroundImg = new
ImageIcon("C:/Users/Eyad/eclipse-workspace/CSC-330/src/com/SpotTheImage/P2.png").getIm
age();
   private Image[] P = new Image[7];
   private NewButton newButton;
   private BackButton backButton;
   private Keyboard qwertyKeyboard;
   private Words words;
   private Word randomWord;
   private char[] guessedLetters;
   private int numOfGuesses;
   private boolean gameOver;

   public MainGamePanel(int WIDTH, int HEIGHT, JPanel container, CardLayout cardLayout,
Words words) {

       this.WIDTH = WIDTH;
       this.HEIGHT = HEIGHT;

       this.words = words;

       this.setLayout(null);

       this.qwertyKeyboard = new Keyboard();

       this.numOfGuesses = 6;

      //displaying our image
       for (int i=0; i<numOfGuesses + 1; i++) {
           Image img = new
ImageIcon("C:/Users/Eyad/eclipse-workspace/CSC-330/src/com/SpotTheImage/P" + i +
".png").getImage();
           this.P[i] = img;
       }

       this.gameOver = false;
```

```
        int buttonWidth = 80;
        int buttonHeight = 30;
        int buttonX = 10;

        String newButtonText = "NEW";
        int newButtonY = 10;
        this.newButton = new NewButton(newButtonText, buttonX, newButtonY, buttonWidth,
buttonHeight);
        this.newButton.addActionListener(this);

        String backButtonText = "BACK";
        int backButtonY = newButtonY + buttonHeight + 10;
        this.backButton = new BackButton(backButtonText, buttonX, backButtonY, buttonWidth,
buttonHeight, container, cardLayout);
        this.backButton.addActionListener(this);

        JPanel keyboard = new JPanel();
        int keyboardWidth = this.WIDTH - 20;
        int keyboardHeight = this.HEIGHT / 2 - 150;
        int keyboardX = 2;
        int keyboardY = this.HEIGHT / 2 + 108;
        keyboard.setBounds(keyboardX, keyboardY, keyboardWidth, keyboardHeight);
        keyboard.setLayout(new GridLayout(3, 9));

        this.qwertyKeyboard.displayButtons(keyboard);
        this.addActionListenerToQwertyKeyboard();

        this.add(this.newButton);
        this.add(this.backButton);
        this.add(keyboard);
    }

    //BELOW HERE IS THE GMAE PART
        @Override
    public void paintComponent(Graphics g) {


        Graphics2D g2D = (Graphics2D) g.create();
        super.paintComponent(g2D);

        char[] wordToChar = this.randomWord.getLettersInWord();
        int space = 10;
        int placeholderLength = this.calculatePlaceholderLength(this.randomWord.getWord(),
space);
```

```
        int x1 = this.calculateX1(placeholderLength, this.randomWord.getWord(), space);
        int y1 = this.HEIGHT / 2 + 90;
        int x2 = x1 + placeholderLength;
        int y2 = y1;
        int fontSize = this.calculatePlaceholderLength(this.randomWord.getWord(), space);
        String fontName = g2D.getFont().getFontName();
        Font font = new Font(fontName, Font.BOLD, fontSize);

        g2D.drawImage(this.backgroundImg, 0, 0, null);



        g2D.setPaint(Color.white);
        g2D.setStroke(new BasicStroke(5));
        g2D.setFont(font);

        this.drawLetterPlaceholders(g2D, wordToChar, x1, y1, x2, y2, space, placeholderLength);

        this.drawGuessedLetters(g2D, x1, y1, x2, space, placeholderLength);



        if (this.gameOver) {
            this.drawGameOverText(g2D);
        }
    }

    private void drawLetterPlaceholders(Graphics2D g2D, char[] wordToChar, int x1, int y1, int x2,
int y2, int space, int placeholderLength) {

        int placeholderX1 = x1;
        int placeholderX2 = x2;
        int placeholderY1 = y1;
        int placeholderY2 = y2;

        for (int i=0; i<wordToChar.length; i++) {
            char ch = wordToChar[i];
            int charX = placeholderX1 + (placeholderLength / 2) -
(g2D.getFontMetrics().stringWidth(String.valueOf(ch).toUpperCase()) / 2);
            int charY = placeholderY1 - 10;

            if (String.valueOf(ch).matches("[a-zA-Z0-9]")) {
                g2D.drawLine(placeholderX1, placeholderY1, placeholderX2, placeholderY2);
            }
            else {
                g2D.drawString(String.valueOf(ch).toUpperCase(), charX, charY);
            }

            placeholderX1 = placeholderX2 + space;
            placeholderX2 = placeholderX1 + placeholderLength;
        }
    }

    private void drawGuessedLetters(Graphics2D g2D, int x1, int y1, int x2, int space, int
placeholderLength) {

        int charX1 = x1;
        int charX2 = x2;

        for (int i=0; i<this.guessedLetters.length; i++) {
            char ch = this.guessedLetters[i];
            int x = charX1 + (placeholderLength / 2) -
(g2D.getFontMetrics().stringWidth(String.valueOf(ch).toUpperCase()) / 2);
            int y = y1 - 10;

            if (String.valueOf(ch).matches("[a-zA-Z0-9]")) {
                g2D.drawString(String.valueOf(ch).toUpperCase(), x, y);
            }
            charX1 = charX2 + space;
            charX2 = charX1 + placeholderLength;
        }
    }
//tool

   private String gameOverText() {

        return this.gameOver && this.guessedWordEqualsWord() ? "The Winner!" : "Game Over!";
    }


    // GAMEOVER!!!!!
    private void drawGameOverText(Graphics2D g2D) {

        String text = this.gameOverText();
        String fontName = g2D.getFont().getFontName();
        int fontSize = 100;
        Font font = new Font(fontName, Font.BOLD, fontSize);
        g2D.setFont(font);
        int textX = (this.WIDTH / 2) - (g2D.getFontMetrics().stringWidth(text) / 2);
        int textY = (this.HEIGHT / 2) - (g2D.getFontMetrics().getHeight() / 2);

        if (text.equals("The Winner!")) {
            g2D.setPaint(Color.GREEN);
        }
        else {
            g2D.setPaint(Color.RED);
        }
        g2D.drawString(text, textX, textY);
    }





    private int calculatePlaceholderLength(String word, int space) {
        /*
        Calculates the length of the letter placeholder based on the screen width, space between
placeholders and the
        length of the word to dynamically adjust the placeholder size to fit the screen.
         */

        int placeholderLength;
        if (word.length() <= 10) {
            placeholderLength = ((this.WIDTH - 40) / 10) - space;
        }
        else {
            placeholderLength = ((this.WIDTH - 20) / word.length()) - space;
        }
        return placeholderLength;
    }

    private int calculateX1(int placeholderLength, String word, int space) {

        int x = (this.WIDTH / 2) - ((placeholderLength * word.length() + space * (word.length() + 1))
/ 2);
        return x;
    }

 private boolean checkLetter(char buttonValue) {

        for (int i=0; i<this.guessedLetters.length; i++) {
            if (buttonValue == this.randomWord.getLettersInWord()[i]) {
                return true;
            }
        }
        return false;
    }

    private void updateLetters(char buttonValue) {

        for (int i=0; i<this.guessedLetters.length; i++) {
            if (buttonValue == this.randomWord.getLettersInWord()[i]) {
                this.guessedLetters[i] = this.randomWord.getLettersInWord()[i];
            }
        }
    }

    private void updateScreen(QwertyButton qwertyButton) {

        boolean letterInWord = this.checkLetter(qwertyButton.getValue());

        if (letterInWord) {
            this.updateLetters(qwertyButton.getValue());
            qwertyButton.setBackground(Color.green);
        }
        else {
            this.numOfGuesses--;
            qwertyButton.setBackground(Color.red);
        }
        qwertyButton.setEnabled(false);
        repaint();
    }

    private void checkForWin() {

  if (this.numOfGuesses == 0) {
                          // Player loses by running out of guesses
                          this.qwertyKeyboard.disableKeyboard();
                          this.gameOver = true;
                        }
                        else if (this.guessedWordEqualsWord()) {
                          // Player wins by guessing the correct word
                          // Select a new random word and reset the game
                          Word word = words.selectRandomWord();
                          word.splitWordToLetters();
```

```
                    this.setRandomWord(word);
                    this.setGuessedLetters(word);
                    this.qwertyKeyboard.resetKeyboard();
                    repaint();
                  }
                }

    private boolean guessedWordEqualsWord() {

        String word = this.randomWord.getWord();
        String guessedWord = String.valueOf(this.guessedLetters);

        return guessedWord.equals(word);
    }

    private void resetGame(ActionEvent e) {

        if (e.getSource() == this.backButton) {
            int result = JOptionPane.showConfirmDialog(this, "Are you sure you want to go back?");
            if (result == 0) {

                this.qwertyKeyboard.resetKeyboard();
                this.backButton.swapCard("2");
            }
        }
        else {

      int result = JOptionPane.showConfirmDialog(this, "Are you sure you want to start a New
Game?");
            if (result == 0) {
                Word word = words.selectRandomWord();
                word.splitWordToLetters();
                this.setRandomWord(word);
                this.setGuessedLetters(word);
                this.qwertyKeyboard.resetKeyboard();
                repaint();
            }
        }
        this.numOfGuesses = 6;
        this.gameOver = false;
    }

    public void setRandomWord(Word word) {
        this.randomWord = word;
    }

    public void setGuessedLetters(Word word) {
        char[] wordToChar = word.getLettersInWord();
        this.guessedLetters = new char[wordToChar.length];

        for (int i=0; i<wordToChar.length; i++) {
            if (!String.valueOf(wordToChar[i]).matches("[a-zA-Z0-9]")) {
                this.guessedLetters[i] = wordToChar[i];
            }
        }
    }

    private void addActionListenerToQwertyKeyboard() {
        for (QwertyButton qwertyButton : this.qwertyKeyboard.getButtons()) {
            qwertyButton.addActionListener(this);
        }
    }

@Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == this.newButton) {
            this.resetGame(e);
        }
        if (e.getSource() == this.backButton) {
            this.resetGame(e);
        }

        for (QwertyButton qwertyButton : this.qwertyKeyboard.getButtons()) {
            if (e.getSource() == qwertyButton) {
                this.updateScreen(qwertyButton);
                this.checkForWin();
            }
        }
    }
}
```

QwertyButton

```
package com.SpotTheImage;

import javax.swing.*;
```

```
public class QwertyButton extends JButton {

        private static final long serialVersionUID = 1L;
        private char value;

   public QwertyButton(char value) {
      this.value = value;

      this.setFocusable(false);
      this.setText(String.valueOf(this.value).toUpperCase());
   }

   public char getValue() {
      return this.value;
   }

}
```

StartScreen

```
package com.SpotTheImage;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class StartScreen extends JPanel implements ActionListener {

   /**
         *
         */
        private static final long serialVersionUID = 1L;
        private PlayButton playButton;
   private ExitButton exitButton;

   public StartScreen(int WIDTH, int HEIGHT, ImageIcon backgroundImg, JPanel container,
CardLayout cardLayout) {

      JLabel startScreenBg = new JLabel(backgroundImg);
```

```
        //below here we creat the dimentions of our buttons
        //button dimensions
        int buttonWidth = 100;
        int buttonHeight = 50;
        int buttonY = HEIGHT - (buttonHeight * 3);

        //PLAY button
        String playButtonText = "PLAY";
        int playButtonX = (WIDTH / 2) - (buttonWidth + 20);
        this.playButton = new PlayButton(playButtonText, playButtonX, buttonY, buttonWidth,
buttonHeight, container, cardLayout);
        this.playButton.addActionListener(this);

        //EXIT button
        String exitButtonText = "EXIT";
        int exitButtonX = (WIDTH / 2) + 20;
        this.exitButton = new ExitButton(exitButtonText, exitButtonX, buttonY, buttonWidth,
buttonHeight);
        this.exitButton.addActionListener(this);

        this.setLayout(new BorderLayout());
        this.add(startScreenBg);

        startScreenBg.setLayout(null);
        startScreenBg.add(this.playButton);
        startScreenBg.add(this.exitButton);
    }

    @Override
    // we have the ability to more home screens, incase we decide to add the feature to give
player the option to select from multiple games
    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == this.playButton) {
            this.playButton.swapCard("2");
        }

        if (e.getSource() == this.exitButton) {
            System.exit(0);
        }
    }
}
```

Word

```
package com.SpotTheImage;

/*
over we Split the words into letters to compare between the letter clicked on the keyboard
 */
public class Word {

private String word;
private char[] lettersInWord;

public Word(String word) {
this.word = word;
}

public void splitWordToLetters() {

this.lettersInWord = this.word.toCharArray();
}

public char[] getLettersInWord() {
return this.lettersInWord;
}

public String getWord() {
return this.word;
}
}
```

Words

```
package com.SpotTheImage;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Random;



public class Words {
```

```
private List<String> wordList; // Add a list to store the different words

private Word emergencyWord;

public Words() {

//list of words
this.wordList = Arrays.asList("cap", "cat", "bug","soccer ball", "baseball", "snake","branch",
"cake", "card","fast food", "paint", "pencil","snack", "basketball", "puzzle","car", "purse", "books");
// Initialize the list with the desired words

new Random();

this.emergencyWord = new Word("Rat");

}


/*
Selects a random word from the list of words
*/
public Word selectRandomWord() {

          if (this.wordList.size() > 0) {
            // Shuffle the list of words
            Collections.shuffle(this.wordList);
            // Return the first word in the shuffled list
            return new Word(this.wordList.get(0));
          }
          return this.emergencyWord;
        }
}
```