import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.\*;  
import javax.swing.Timer;  
import java.util.List;  
import java.io.IOException;  
import javax.imageio.ImageIO;  
import java.awt.image.BufferedImage;  
  
public class ClueGuest extends JFrame {  
 private JLabel wordLabel, guessesLabel, messageLabel, scoreLabel, categoryLabel, clueLabel, playerNameLabel, timerLabel;  
 private JTextField guessField;  
 private JButton guessButton, restartButton, quitButton, submitWordButton;  
 private List<String> categories = Arrays.asList("Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Place in the Philippines", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Filipino dish", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home", "Things found in the home");  
 private List<String> clues = Arrays.asList("Capital city", "Queen City of the South", "Durian capital", "Largest city by area", "Summer capital", "White sand beaches", "Underground River", "Chocolate Hills", "Taal Volcano view", "Dive sites", "City of Love", "City of Smiles", "Heritage city", "Freeport Zone", "Entertainment hub", "Latin City", "Mayon Volcano view", "University town", "Yolanda-hit city", "Honda Bay", "Pilgrim city", "City of Golden Friendship", "Surfing capital", "Orchid city", "Balangay boats", "Lake Lanao view", "Cagayan River", "City of Majestic Waterfalls", "Coconut city", "Seafood capital", "Pineapple capital", "World War II sites", "Tuna capital", "La Union surf", "Windmills view", "Wakeboarding capital", "Coconut pie", "Sugar capital", "Marbel capital", "Longest bridge", "Bicol express", "Rice granary", "Rice bowl", "Pineapple hub", "Charter Day celebration", "Whaleshark interaction", "Grand Mosque", "Cotta For", "Water falls chase", "Dakak Beach", "Vinegar and soy sauce", "Sour soup", "Roast pig", "Peanut sauce stew", "Fried spring rolls", "Stir-fried noodles", "Ginger chicken soup", "Sour soup", "Spicy coconut dish", "Grilled chicken", "Mixed dessert", "Sizzling pork", "Duck embryo", "Beef with garlic rice and egg", "Rice cake", "Steamed rice cake", "Mixed vegetable stew", "Blood stew", "Soybean pudding", "Beef marrow soup", "Purple rice cake", "Taro leaves in coconut milk", "Filipino sausage", "Chocolate rice porridge", "Chicken rice porridge", "Eggplant omelette", "Fermented shrimp paste", "Vegetable stew in coconut milk", "Chicken in coconut milk", "Chicken in tamarind broth", "Garlic fried rice", "Sticky rice dessert", "Rice flour delicacy", "Shrimp fritters", "Sweet cured pork", "Caramel custard", "Eggplant salad", "Grilled squid", "Filipino ceviche", "Vegetable soup", "Pineapple chicken stew", "Sweet rice balls in coconut milk", "Seafood noodle dish", "Peanut sauce stew", "Ilocano pork dish", "Chicken stew with tomato sauce", "Jackfruit in coconut milk", "Creamy chicken soup", "Ground meat dish", "Noodle dish from Lucban", "You wash your dishes in me", "You sleep on me every night", "You keep your food cold in me", "You sit on me at the dining table", "You hang your clothes on me", "You keep your books on me", "You dry your clothes in me", "You bake delicious treats in me", "You display family photos in me", "You light up the room with me", "I'm a fluffy cushion for your head", "You sit on me while watching TV", "You use me to clean the floors", "I hold your precious memories", "I keep your clothes clean", "I keep the sun out of your room", "I keep your food fresh", "You sleep under me", "I hold your favorite books", "You use me to brush your teeth", "I keep your clothes wrinkle-free", "I keep your papers organized", "I keep your plants alive", "You use me to clean the dishes", "I hold your dirty laundry", "You use me to close windows and doors", "You use me to measure ingredients", "I keep your drinks hot", "You use me to type on the computer", "I keep your clock ticking", "You use me to open cans","You use me to sweep the floors", "Used to store leftovers in the fridge ", "Used to chop ingredients while cooking", "Used to cook on the stove", "Used to ring in important calls", "Used to keep the house cool", "Used to control the lights", "Used to chop vegetables", "Keeps your trash contained", "Where you store your extra blankets and pillows", "Used to heat up water for tea or coffee", "Used to toast bread", "You write with me", "You use me to style your hair", "You use me to see in the dark", "You use me to cook food evenly ", "You use me to mix ingredients ", "You use me to hold your phone ", "You use me to carry groceries");  
 private List<List<String>> wordLists = Arrays.asList(  
 Arrays.asList("MANILA"),  
 Arrays.asList("CEBU"),   
 Arrays.asList("DAVAO"),   
 Arrays.asList("QUEZON"),   
 Arrays.asList("BAGUIO"),   
 Arrays.asList("BORACAY"),   
 Arrays.asList("PALAWAN"),   
 Arrays.asList("BOHOL"),  
 Arrays.asList("TAGAYTAY"),   
 Arrays.asList("BATANGAS"),   
 Arrays.asList("ILOILO"),   
 Arrays.asList("BACOLOD"),   
 Arrays.asList("VIGAN"),   
 Arrays.asList("SUBIC"),   
 Arrays.asList("ANGELES"),  
 Arrays.asList("ZAMBOANGA"),   
 Arrays.asList("LEGAZPI"),   
 Arrays.asList("DUMAGUETE"),   
 Arrays.asList("TACLOBAN"),   
 Arrays.asList("PUERTOPRINCESA"),   
 Arrays.asList("NAGA"),  
 Arrays.asList("CAGAYAN"),   
 Arrays.asList("SURIGAO"),   
 Arrays.asList("DIPOLOG"),   
 Arrays.asList("BUTUAN"),   
 Arrays.asList("MARAWI"),   
 Arrays.asList("TUGUEGARAO"),   
 Arrays.asList("ILIGAN"),  
 Arrays.asList("CALBAYOG"),   
 Arrays.asList("ROXAS"),   
 Arrays.asList("MALAYBALAY"),   
 Arrays.asList("BATAAN"),   
 Arrays.asList("GENERALSANTOS"),   
 Arrays.asList("SANFERNANDO"),  
 Arrays.asList("LAOAG"),   
 Arrays.asList("NAGA"),   
 Arrays.asList("LUCENA"),   
 Arrays.asList("TARLAC"),   
 Arrays.asList("KORONADAL"),   
 Arrays.asList("BAYBAY"),   
 Arrays.asList("LEGAZPI"),  
 Arrays.asList("SANCARLOS"),   
 Arrays.asList("CABANATUAN"),   
 Arrays.asList("ORMOC"),   
 Arrays.asList("BAYUGAN"),   
 Arrays.asList("SORSOGON"),   
 Arrays.asList("COTABATO"),   
 Arrays.asList("OZAMIZ"),  
 Arrays.asList("TANDAG"),   
 Arrays.asList("PAGADIAN"),  
 Arrays.asList("ADOBO"),  
 Arrays.asList("SINIGANG"),  
 Arrays.asList("LECHON"),   
 Arrays.asList("KARE-KARE"),   
 Arrays.asList("LUMPIA"),   
 Arrays.asList("PANCIT"),   
 Arrays.asList("TINOLA"),   
 Arrays.asList("SINIGANG"),   
 Arrays.asList("BICOLEXPRESS"),  
 Arrays.asList("CHICKENINASAL"),   
 Arrays.asList("HALO-HALO"),   
 Arrays.asList("SISIG"),   
 Arrays.asList("BALUT"),   
 Arrays.asList("TAPSILOG"),  
 Arrays.asList("BIBINGKA"),  
 Arrays.asList("PUTO"),  
 Arrays.asList("PINAKBET"),  
 Arrays.asList("DINUGUAN"),  
 Arrays.asList("TAHO"),  
 Arrays.asList("BULALO"),  
 Arrays.asList("PUTOBUMBONG"),  
 Arrays.asList("LAING"),   
 Arrays.asList("LONGGANISA"),  
 Arrays.asList("CHAMPORADO"),   
 Arrays.asList("ARROZCALDO"),   
 Arrays.asList("TORTANGTALONG"),   
 Arrays.asList("BAGOONG"),   
 Arrays.asList("GINATAANGGULAY"),   
 Arrays.asList("GINATAANGMANOK"),  
 Arrays.asList("SINAMPALUKANGMANOK"),   
 Arrays.asList("SINANGAG"),   
 Arrays.asList("SUMAN"),   
 Arrays.asList("ESPASOL"),   
 Arrays.asList("UKOY"),   
 Arrays.asList("TOCINO"),   
 Arrays.asList("LECHEFLAN"),  
 Arrays.asList("ENSALADANGTALONG"),   
 Arrays.asList("INIHAWNAPUSIT"),   
 Arrays.asList("KINILAW"),   
 Arrays.asList("BULANGLANG"),   
 Arrays.asList("PININYAHANGMANOK"),   
 Arrays.asList("GINATAANGBILO-BILO"),  
 Arrays.asList("PANCITMALABON"),   
 Arrays.asList("KARE-KARE"),   
 Arrays.asList("DINAKDKAN"),   
 Arrays.asList("CHICKENAFRITADA"),   
 Arrays.asList("GINATAANGLANGKA"),   
 Arrays.asList("SOPAS"),   
 Arrays.asList("GINILING"),  
 Arrays.asList("PANCITHABHAB"),  
 Arrays.asList("SINK"),  
 Arrays.asList("BED"),  
 Arrays.asList("FRIDGE"),  
 Arrays.asList("CHAIR"),  
 Arrays.asList("CLOSET"),  
 Arrays.asList("SHELF"),  
 Arrays.asList("DRYER"),  
 Arrays.asList("OVEN"),  
 Arrays.asList("PICTUREFRAME"),  
 Arrays.asList("LAMP"),  
 Arrays.asList("PILLOW"),  
 Arrays.asList("SOFA"),  
 Arrays.asList("BROOM"),  
 Arrays.asList("PHOTOALBUM"),  
 Arrays.asList("WASHINGMACHINE"),  
 Arrays.asList("CURTAINS"),  
 Arrays.asList("PANTRY"),  
 Arrays.asList("BLANKET"),  
 Arrays.asList("BOOKSHELF"),  
 Arrays.asList("TOOTHBRUSHHOLDER"),  
 Arrays.asList("IRONINGBOARD"),  
 Arrays.asList("DESKORGANIZER"),  
 Arrays.asList("WATERINGCAN"),  
 Arrays.asList("DISHWASHINGLIQUID"),  
 Arrays.asList("LAUNDRYBASKET"),  
 Arrays.asList("LOCK"),  
 Arrays.asList("MEASURINGCUP"),  
 Arrays.asList("THERMOS"),  
 Arrays.asList("KEYBOARD"),  
 Arrays.asList("BATTERY"),  
 Arrays.asList("CANOPENER"),  
 Arrays.asList("DUSTPAN"),  
 Arrays.asList("TUPPERWARE"),  
 Arrays.asList("CUTTINGBOARD"),  
 Arrays.asList("POT"),  
 Arrays.asList("PHONE"),  
 Arrays.asList("FAN"),  
 Arrays.asList("SWITCH"),  
 Arrays.asList("KNIFE"),  
 Arrays.asList("GARBAGEBIN"),  
 Arrays.asList("STORAGEBOX"),  
 Arrays.asList("KETTLE"),  
 Arrays.asList("TOASTER"),  
 Arrays.asList("PEN"),  
 Arrays.asList("COMB"),  
 Arrays.asList("FLASHLIGHT"),  
 Arrays.asList("SPATULA"),  
 Arrays.asList("MIXINGBOWL"),  
 Arrays.asList("PHONECASE"),  
 Arrays.asList("SHOPPINGBAG")   
   
 );  
 private String currentWord, currentCategory, currentClue, playerName;  
 private List<Character> guessedLetters;  
 private int incorrectGuesses, maxIncorrectGuesses = 6, score, currentWordIndex, timeElapsed = 0;  
 private Timer timer;  
 private static Map<String, Integer> leaderboard = new TreeMap<>();  
 private BufferedImage backgroundImage;  
  
 public ClueGuest(String playerName) {  
 this.playerName = playerName;  
 setTitle("Clue Guest");  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 setExtendedState(JFrame.MAXIMIZED\_BOTH);  
 setLocationRelativeTo(null);  
   
 // Load the background image  
  
 try {  
 backgroundImage = ImageIO.read(getClass().getResource("background.jpg")); // Change "background.jpg" to your image file name  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
   
 // Initialize GUI components  
 wordLabel = new JLabel();  
 wordLabel.setFont(new Font("Arial", Font.BOLD, 50));  
 wordLabel.setForeground(Color.WHITE); // Set text color to white  
 guessesLabel = new JLabel();  
 guessesLabel.setFont(new Font("Arial", Font.PLAIN, 30));  
 guessesLabel.setForeground(Color.WHITE); // Set text color to white  
 messageLabel = new JLabel();  
 messageLabel.setFont(new Font("Arial", Font.PLAIN, 20));  
 messageLabel.setForeground(Color.WHITE); // Set text color to white  
   
 // Modify JTextField appearance  
 guessField = new JTextField(5);  
 guessField.setPreferredSize(new Dimension(400, 30)); // Adjust the width and height as needed  
 guessField.setFont(new Font("Arial", Font.PLAIN, 20));  
 guessField.setBorder(BorderFactory.createCompoundBorder(  
 BorderFactory.createLineBorder(Color.BLACK, 0), // blackS border with thickness 0  
 BorderFactory.createEmptyBorder(5, 10, 5, 10) // Empty border with padding  
 ));  
 guessField.setForeground(Color.BLACK); // Set text color to black  
 guessButton = new JButton("Guess");  
 guessButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 processGuess();  
 }  
 });  
 guessButton.setForeground(Color.BLACK); // Set text color to black  
 restartButton = new JButton("Next");  
 restartButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (getCurrentWordState().replaceAll(" ", "").equals(currentWord)) {  
 startNewGame();  
 } else {  
 JOptionPane.showMessageDialog(ClueGuest.this, "Please guess the current word before proceeding to the next one.", "Incomplete Guess", JOptionPane.WARNING\_MESSAGE);  
 }  
 }  
 });  
 restartButton.setForeground(Color.BLACK); // Set text color to black  
 quitButton = new JButton("Quit");  
 quitButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 System.exit(0);  
 }  
 });  
 quitButton.setForeground(Color.BLACK); // Set text color to black  
 submitWordButton = new JButton("Submit");  
 submitWordButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 submitWord();  
 }  
 });  
   
 // Modify JButton sizes  
 guessButton.setPreferredSize(new Dimension(110, 40)); // Adjust width and height as needed  
 restartButton.setPreferredSize(new Dimension(110, 40)); // Adjust width and height as needed  
 quitButton.setPreferredSize(new Dimension(110, 40)); // Adjust width and height as needed  
 submitWordButton.setPreferredSize(new Dimension(110, 40)); // Adjust width and height as needed   
 submitWordButton.setForeground(Color.BLACK); // Set text color to black  
 scoreLabel = new JLabel("Score: 0");  
 scoreLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Adjust font size  
   
   
 submitWordButton.setFont(new Font("Arial", Font.BOLD, 16)); // Change the font size as needed  
 guessButton.setFont(new Font("Arial", Font.BOLD, 16)); // Change the font size as needed  
 restartButton.setFont(new Font("Arial", Font.BOLD, 16)); // Change the font size as needed  
 quitButton.setFont(new Font("Arial", Font.BOLD, 16)); // Change the font size as needed   
   
   
 scoreLabel.setForeground(Color.WHITE); // Set text color to white  
 categoryLabel = new JLabel("Category: ");  
 categoryLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Adjust font size  
 categoryLabel.setForeground(Color.WHITE); // Set text color to white  
 clueLabel = new JLabel("Clue: ");  
 clueLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Adjust font size  
 clueLabel.setForeground(Color.WHITE); // Set text color to white  
 playerNameLabel = new JLabel("Player: " + playerName);  
 playerNameLabel.setFont(new Font("Arial", Font.BOLD, 20)); // Adjust font size  
 playerNameLabel.setForeground(Color.WHITE); // Set text color to white  
 timerLabel = new JLabel("Time: 0");  
 timerLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Adjust font size  
 timerLabel.setForeground(Color.WHITE); // Set text color to white  
  
 // Create JPanel and add components  
 JPanel panel = new JPanel(new GridBagLayout()) {  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 if (backgroundImage != null) {  
 g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);  
 }  
 }  
 };  
 panel.setForeground(Color.WHITE); // Set text color to white  
 GridBagConstraints constraints = new GridBagConstraints();  
 constraints.gridx = 0;  
 constraints.gridy = 0;  
 constraints.insets = new Insets(10, 10, 10, 10);  
 panel.add(playerNameLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(wordLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(guessesLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(messageLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(guessField, constraints);  
  
 constraints.gridy++;  
 panel.add(guessButton, constraints);  
  
 constraints.gridx = 1;  
 panel.add(restartButton, constraints);  
  
 constraints.gridx = 2;  
 panel.add(submitWordButton, constraints);  
  
 constraints.gridx = 3;  
 panel.add(quitButton, constraints);  
  
 constraints.gridx = 0;  
 constraints.gridy++;  
 panel.add(scoreLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(categoryLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(clueLabel, constraints);  
  
 constraints.gridy++;  
 panel.add(timerLabel, constraints);  
  
 add(panel);  
   
 startNewGame();  
  
 timer = new Timer(1000, new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 timeElapsed++;  
 timerLabel.setText("Time: " + timeElapsed + " sec");  
  
 if (timeElapsed >= 60) {  
 timer.stop();  
 displayGameOverDialog();  
 }  
 }  
 });  
 timer.start();  
 }  
  
 private void startNewGame() {  
 currentWordIndex = new Random().nextInt(categories.size());  
 currentCategory = categories.get(currentWordIndex);  
 currentClue = clues.get(currentWordIndex);  
 List<String> selectedWordList = wordLists.get(currentWordIndex);  
 currentWord = selectedWordList.get(new Random().nextInt(selectedWordList.size()));  
 guessedLetters = new ArrayList<>();  
 incorrectGuesses = 0;  
 updateUI();  
 }  
  
 private void processGuess() {  
 char guess = guessField.getText().toUpperCase().charAt(0);  
 guessField.setText("");  
  
 if (guessedLetters.contains(guess)) {  
 messageLabel.setText("You already guessed that letter. Try again.");  
 return;  
 }  
 guessedLetters.add(guess);  
  
 if (currentWord.indexOf(guess) == -1) {  
 incorrectGuesses++;  
 score -= 5;  
 messageLabel.setText("Incorrect guess. Try again.");  
 } else {  
 messageLabel.setText("Correct guess!");  
 score += 10;  
 }  
 scoreLabel.setText("Score: " + score);  
 updateUI();  
  
 if (incorrectGuesses >= maxIncorrectGuesses) {  
 messageLabel.setText("Game over! The word was " + currentWord);  
 timer.stop();  
 displayGameOverDialog();  
 } else if (getCurrentWordState().replaceAll(" ", "").equals(currentWord)) {  
 messageLabel.setText("Congratulations, you guessed the word!");  
 timer.stop();  
 score += currentWord.length() \* 10;  
 currentWordIndex = (currentWordIndex + 1) % categories.size();  
 startNewGame();  
 timer.start();  
 }  
 }  
  
 private void submitWord() {  
 String submittedWord = JOptionPane.showInputDialog(this, "Enter the correct word:");  
 if (submittedWord != null) {  
 submittedWord = submittedWord.toUpperCase();  
 if (submittedWord.equals(currentWord)) {  
 messageLabel.setText("Correct! You guessed the word.");  
 int currentWordPoints = submittedWord.length() \* 10;  
 score += currentWordPoints;  
 scoreLabel.setText("Score: " + score);  
 currentWordIndex = (currentWordIndex + 1) % categories.size();  
 startNewGame();  
 } else {  
 messageLabel.setText("Incorrect word. Game over!");  
 score -= 5;  
 scoreLabel.setText("Score: " + score);  
 timer.stop();  
 displayGameOverDialog();  
 }  
 }  
 }  
  
 private void updateUI() {  
 wordLabel.setText(getCurrentWordState());  
 guessesLabel.setText("Incorrect guesses: " + incorrectGuesses + "/" + maxIncorrectGuesses);  
 categoryLabel.setText("Category: " + currentCategory);  
 clueLabel.setText("Clue: " + currentClue);  
 }  
  
 private String getCurrentWordState() {  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < currentWord.length(); i++) {  
 char c = currentWord.charAt(i);  
 if (guessedLetters.contains(c)) {  
 sb.append(c);  
 } else {  
 sb.append("\_");  
 }  
 sb.append(" ");  
 }  
 return sb.toString().trim();  
 }  
  
 private void displayGameOverDialog() {  
 int option = JOptionPane.showConfirmDialog(this,  
 "Game Over!\nPlayer: " + playerName + "\nScore: " + score + "\nDo you want to play again?",  
 "Game Over",  
 JOptionPane.YES\_NO\_OPTION);  
  
 if (option == JOptionPane.YES\_OPTION) {  
 score = 0;  
 timeElapsed = 0;  
 timerLabel.setText("Time: 0");  
 startNewGame();  
 timer.start();  
 } else {  
 leaderboard.put(playerName, score);  
 leaderboard = sortByValue(leaderboard);  
 displayLeaderboard();  
 }  
 }  
  
 private Map<String, Integer> sortByValue(Map<String, Integer> unsortedMap) {  
 List<Map.Entry<String, Integer>> list = new LinkedList<>(unsortedMap.entrySet());  
 list.sort((o1, o2) -> (o2.getValue()).compareTo(o1.getValue()));  
  
 Map<String, Integer> sortedMap = new LinkedHashMap<>();  
 for (Map.Entry<String, Integer> entry : list) {  
 sortedMap.put(entry.getKey(), entry.getValue());  
 }  
 return sortedMap;  
 }  
  
 private void displayLeaderboard() {  
 StringBuilder leaderboardText = new StringBuilder("Leaderboard:\n");  
 int rank = 1;  
 for (Map.Entry<String, Integer> entry : leaderboard.entrySet()) {  
 leaderboardText.append(rank).append(". ").append(entry.getKey()).append(": ").append(entry.getValue()).append("\n");  
 rank++;  
 }  
 JOptionPane.showMessageDialog(this, leaderboardText.toString(), "Leaderboard", JOptionPane.INFORMATION\_MESSAGE);  
 startNewPlayerGame();  
 }  
  
 private void startNewPlayerGame() {  
 playerName = JOptionPane.showInputDialog(null, "Enter your name:");  
 if (playerName != null && !playerName.isEmpty()) {  
 playerNameLabel.setText("Player: " + playerName);  
 score = 0;  
 timeElapsed = 0;  
 timerLabel.setText("Time: 0");  
 startNewGame();  
 timer.start();  
 } else {  
 JOptionPane.showMessageDialog(null, "Please enter a valid name.", "Error", JOptionPane.ERROR\_MESSAGE);  
 startNewPlayerGame();  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(new Runnable() {  
 public void run() {  
 String playerName = JOptionPane.showInputDialog(null, "Enter your name:");  
 if (playerName != null && !playerName.isEmpty()) {  
 ClueGuest clueGuest = new ClueGuest(playerName);  
 clueGuest.setVisible(true);  
 } else {  
 JOptionPane.showMessageDialog(null, "Please enter a valid name.", "Error", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 });  
 }  
}