```
**Application Purpose: This is the main class for the game "Word Guessing"
 * Author: Bibek Poudel
 * Date: 12 April 2021
 * Time: 11:30 am
 */

//importing the Scanner from the util package to scan the player's input
import java.util.Scanner;

//the game class which contains the main method
public class Game {

   //the main method
   public static void main(String[] args) {

      //Requesting player's for their name
      System.out.println("Enter your name:");
      //initializing in for scanning name
      Scanner in = new Scanner(System.in);
      //creating gamer object to store the player's information
      Player gamer = new Player();
      //using setters to set player's name
      gamer.setPlayerName(in.nextLine());

      //Welcome message with the name of player
      System.out.println("Welcome to the word guessing game, " + gamer.getPlayerName());

      //while loop to continue the game until the player's wants (playGame == true) to be with game
      boolean playGame = true;
```

```
while(playGame){

   //Choices to choose levels
   System.out.println("Choose the level of difficulty for the game:");
   System.out.println("Enter '1' for Easy");
   System.out.println(("Enter '2' for Medium"));
   System.out.println("Enter '3' for Hard");

   //initializing gameLevel object with new Level()
   Level gameLevel = new Level();

   //initializing variable to store level
   int level;

   //exception handling using try catch to deal with the error caused by wrong level input
   try{
      //storing the level chose by player
      level = in.nextInt();
   }
   catch (java.util.InputMismatchException e)
   {
      //re-prompts the user for level input
      System.out.println("Input the numeric value between 1-3");
      Scanner input = new Scanner(System.in);
      level = input.nextInt();
      input.close();
   }
   //if the numeric level input is still not matching with the levels
   //it again re-prompts user for input until the level is matched
```

```
        while (!Player.validatePlayerInput(level))
        {
           System.out.println("Give the numeric value between 1 to 3");
           Scanner put = new Scanner(System.in);
           level = put.nextInt();
        }
        //runs the level chosen by the player
        gameLevel.runlevel(level);

        //decides if the user wants to play the new game again
        playGame = gamer.playAgain();
      }

      //Displays the game over message after the game ends
      System.out.println("Game Over, "+ gamer.getPlayerName());
   }
}
/**Application Purpose: To create a class object to store the words to display as a question
 * Author: Bibek Poudel
 * Date: 12 April 2021
 * Time: 11:30 am
 */

//this class contains words as well as methods to pass words to other classes
public class WordList {

   //single dimension array containing list of words to display as a question to players
   private String []questionWords = {"PET","RED", "GOD", "RAM", "CUT",
                              "BANG", "FlAG", "BODY", "GAME", "MODE",
```

```
                "RANDOM", "OBJECT", "LINKED", "IMPORT", "BEFORE"};

    //this method returns list of words depending on the length ordered
    public String[] giveQuestionWords(int length)
    {
        //the single dimension array word list contains 5 words
        String []wordList = new String[5];
        //this stores the  index where the words should be stored
        int indexOfWordList = 0;

        //this loop search the appropriate words in the array and stores it return list
        for(int i = 0; i < questionWords.length; i++)
        {
            //checks of the required length is matched
            if(questionWords[i].length() == length)
            {
                //stores words if matched
                wordList[indexOfWordList] = questionWords[i];
                //increases the index to store next word
                indexOfWordList++;
            }
        }
        //returns word list
        return wordList;
    }
}
/**Application Purpose: To create a level class which contains the information each level requires in game
 * Author: Bibek Poudel
```

```
 * Date: 12 April 2021
 * Time: 11:30 am
 */

//importing random for generating the random numbers
import java.util.Random;
//importing scanner to scan user's input
import java.util.Scanner;

//this is Level class
public class Level {

   //a multi dimension array to store the record of the answers (true/false) and the question number
   private String [][]answerRecord =new String[5][2];

   //calling this method runs the level selected by the user
   public void runlevel(int level)
   {
      //initializing WordList object as list
      WordList list = new WordList();
      //declaring a new string array to store the 5 question words for each level
      String [] words= new String[5];

      //this series of if statements generates the words list depending on level
      if(level == 1)
      {
         //3 letter words for level 1
         words = list.giveQuestionWords(3);
      }
```

```
if(level == 2)
{
   //four letter words for level 2
   words = list.giveQuestionWords(4);
}
if(level == 3)
{
   //6 letter words for level 3
   words = list.giveQuestionWords(6);
}
//this loop continues until the words contained in each levels are all answered
for(int i= 0; i < 5; i++)
{
   //this will store the number of question word displayed
   answerRecord[i][0] = (i + 1) + " = ";

   //displaying the message to show how many words out of total are displayed
   System.out.println("Guess the word: ("+ (1+i) + "/5)");

   //this array stores the position in the word where blank spaces are contained
   int []blankPositions = generateBlankPosition(level, words[i].length());

   //wordsToDisplay stores words to display as a question
   String wordToDisplay = words[i];
   //this statements checks level before generating the question words
   if(level == 1)
   {
      //question words contains 1 blank space to be filled for level 1
      wordToDisplay = generateWordsToDisplay(words[i], blankPositions[0]);
```

```
        }
        if(level == 2)
        {
           //question words contains 2 blank space to be filled for level 2
           wordToDisplay = generateWordsToDisplay(words[i], blankPositions[0], blankPositions[1]);
        }
        if(level == 3)
        {
           //question words contains 3 blank space to be filled for level 3
           wordToDisplay = generateWordsToDisplay(words[i], blankPositions[0], blankPositions[1],
blankPositions[2]);
        }

        //scanner for scanning user's answer
        Scanner in = new Scanner(System.in);
        String answer; //to store the answer from user

        //do-while loop gives chances for the user to input until the life is over or word is guessed
        do{
           //displays question word to user
           System.out.println(wordToDisplay);

           //answer stores the answer fetched by player
           answer = in.nextLine();
           //converting answer into uppercase for comparison using string class method
           answer = answer.toUpperCase();

           //it stores true if the guessed letter is matched, false if not matched
           boolean matched = false;
```

```
        //using switch to check the word guessed are matched or not
        switch (level)
        {
           /*for level 3 there are 3 blank space so the answer is checked with 3 three values replaced by blank
            space. So, if there is no break after each case */
           case 3:
              //checks if the letter guessed is matched with the words in third blank space.
              //Wrapper class method String.valueOf to return the string representation of the char argument
              if(answer.equalsIgnoreCase(String.valueOf(words[i].charAt(blankPositions[2]))))
              {
                 //replaces the question word by the correct letter if matched
                 wordToDisplay = wordToDisplay.substring(0,blankPositions[2]) + answer + wordToDisplay.substring(blankPositions[2] +1);
                 matched = true;
              }
           case 2:
              //checks if the letter guessed is matched with the words in second blank space
              //Wrapper class method String.valueOf to return the string representation of the char argument
              if(answer.equalsIgnoreCase(String.valueOf(words[i].charAt(blankPositions[1]))))
              {
                 //replaces the question word by the correct letter if matched
                 wordToDisplay = wordToDisplay.substring(0,blankPositions[1]) + answer + wordToDisplay.substring(blankPositions[1] +1);
                 matched = true;
              }
           case 1:
```

```
                //checks if the letter guessed is matched with the words in first blank space
                //Wrapper class method String.valueOf to return the string representation of the char argument
                if (answer.equalsIgnoreCase(String.valueOf(words[i].charAt(blankPositions[0]))))
                {
                   //replaces the question word by the correct letter if matched
                   wordToDisplay = wordToDisplay.substring(0,blankPositions[0]) + answer + wordToDisplay.substring(blankPositions[0] +1);
                   matched = true;
                }
             default:
                //if there is no match found the life of player will decrease by one and shows the message
                if(matched == false)
                {
                   Player.playerLife--;
                   System.out.println("Incorrect Word");
                   System.out.println(Player.playerLife +" chance Remaining.");
                }
          }
          //before next loop it checks if the player life is over or if the word is guessed
       }while (Player.playerLife > 0 && wordToDisplay.contains("_"));

       //if user's life is over
       if(Player.playerLife == 0)
       {
          //prints the message
          System.out.println("You failed to guess the word: " + words[i]);
          //stores that the answer is false for this word
          answerRecord[i][1] = "false ";
```

```
        }
        //else if the player guessed the word before the life ended
        else{
            //stores that the answer is true for this word
            answerRecord[i][1] = "true| ";
            //displays the word guessed
            System.out.println(wordToDisplay);
            //prints the message
            System.out.println("You complete the word, Keep it up!");
        }
        //printing a line before next word is shown
        System.out.println("-----------------------------------------------------------------------------------");
        //sets the player life to the default value for nex time
        Player.playerLife = 10;
    }
    //Displaying the answerRecord after completing the level
    System.out.println("Your all answers are:");
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 2; j++) {
            System.out.print(answerRecord[i][j]);
        }
    }
    System.out.println();
    //printing a line before next word is shown
    System.out.println("-----------------------------------------------------------------------------------");
}

//this method generates blank position for each word depending on the level and length of word
public int[] generateBlankPosition(int level, int length)
```

```
{
   //array to store position the position of blank spaces
   //level 3 has 3 blank position
   //level 2 has 2 blank position
   //level 1 has 1 blank position
   int[] blankSpacePositions = new int[level];

   //initializing randomNumber
   Random randomNumber = new Random();

   //this loop repeats depending on the number of blank spaces needed in a level
   for (int i = 0; i< level; i++)
   {
      //stores a random blank space position
      blankSpacePositions[i] = randomNumber.nextInt(length);

      //this loop to check the new number with the other number previously stored
      for(int j = 0; j < i; j++)
      {
         //if the number is repeated before it will generate new unique number
         while(blankSpacePositions[i]==blankSpacePositions[j])
         {
            blankSpacePositions[i] = randomNumber.nextInt(length);
         }
      }
   }
   //returns the array storing the positions for blank spaces
   return blankSpacePositions;
}
```

```
   //these are overloaded methods which generated the question words to display depending on the
number of blank spaces


   public String generateWordsToDisplay(String words, int firstBlank)
   {
      String wordsToDisplay = words;
      //replace one of the letter of the word by blank space
      wordsToDisplay = wordsToDisplay.replaceAll(wordsToDisplay.substring(firstBlank,firstBlank+1),"_");


      //returns word with 1 blank spaces to be guessed
      return wordsToDisplay;
   }
   public String generateWordsToDisplay(String words, int firstBlank, int secondBlank)
   {
      String wordsToDisplay = words;
      //replace one of the letter of the word by blank space
      wordsToDisplay = wordsToDisplay.replaceAll(wordsToDisplay.substring(firstBlank,firstBlank+1),"_");
      //replace next one of the letter of the word by blank space
      wordsToDisplay =
wordsToDisplay.replaceAll(wordsToDisplay.substring(secondBlank,secondBlank+1),"_");


      //returns word with 2 blank spaces to be guessed
      return wordsToDisplay;
   }
   public String generateWordsToDisplay(String words, int firstBlank, int secondBlank, int thirdBlank)
   {
      String wordsToDisplay = words;
      //replace one of the letter of the word by blank space
```

```
        wordsToDisplay = wordsToDisplay.replaceAll(wordsToDisplay.substring(firstBlank,firstBlank+1),"_");
        //replace next one of the letter of the word by blank space
        wordsToDisplay =
wordsToDisplay.replaceAll(wordsToDisplay.substring(secondBlank,secondBlank+1),"_");
        //replace next one of the letter of the word by blank space
        wordsToDisplay =
wordsToDisplay.replaceAll(wordsToDisplay.substring(thirdBlank,thirdBlank+1),"_");

        //returns word with 3 blank spaces to be guessed
        return wordsToDisplay;
    }
}
/**Application Purpose: To create a level class which contains the information a player
 * Author: Bibek Poudel
 * Date: 12 April 2021
 * Time: 11:30 am
 */

//importing scanner to take user's input
import java.util.Scanner;

//this is player class
public class Player {

    //instance variable which store the player's name
    private String playerName;

    //class variable to store the players life
    public  static int playerLife = 10;
```

```
   //setter method to set the player's name
   public void setPlayerName(String playerName) {
      this.playerName = playerName;
   }

   //getter method to return player name
   public String getPlayerName() {
      return playerName;
   }

   //a static method which checks if the level chosen by player is valid or not
   public static boolean validatePlayerInput(int choseLevel)
   {
      //stores the result in isValidated
      boolean isValidated = false;

      //if choseLevel is matched the isValidated world be true and an appropriate message would be
displayed
      if(choseLevel == 1 || choseLevel == 2 || choseLevel == 3)
      {
         isValidated = true;
         //switch statement to display message depending on the level chosen
         switch (choseLevel)
         {
            case 1:
               System.out.println("You chose easy level");
               break;
            case 2:
```

```
            System.out.println("You chose medium level");
            break;
        case 3:
            System.out.println("You chose difficult level");
            break;
      }
      System.out.println("Let's start the game.");
   }
   //returns the result
   return isValidated;
}

//this method asks players to play again and returns the decision made by player
public boolean playAgain()
{
   //isPlayingAgain stores the decision made by player
   boolean isPlayingAgain = false;

   //isInputValid keeps record that the input is valid or not
   boolean isInputValid = false;

   //scanner to scan user's input
   Scanner input = new Scanner(System.in);

   //this do while loop repeats until the the inout is valid
   do{
      //asks the player if they want to play again
      System.out.println(getPlayerName() + ", Do you wanna play again? (yes/no)");
      //decision stores their result
```

```
        String decision = input.nextLine();

        //these if statements checks if the input matched with the expected input
        //if input is yes or y
        if(decision.equalsIgnoreCase("yes") || decision.equalsIgnoreCase("y"))
        {
           //it is true that player would be playing again
           isPlayingAgain  = true;
           //it is true that input is valid
           isInputValid = true;
        }
        //if input is no or n
        if(decision.equalsIgnoreCase("no") || decision.equalsIgnoreCase("n"))
        {
           isPlayingAgain = false;
           isInputValid = true;
        }
        //while checks if the input is valid or not before terminating the loop
      }while(!isInputValid);

      //returns whether the player is playing again or not
     return isPlayingAgain;
   }
}
```