Name	**Virinchi Sadashiv Shettigar**
UID no.	**2021300118**
Experiment No.	**8**

AIM:	**Program on Abstraction: Implement a Program to demonstarte Abstraction using abstract class**
Program 1	
PROBLEM STATEMENT:	**Write a program that plays the game of hangman. In hangman, the computer begins by selecting a secret word at random from a list of possibilities. It then prints out a row of dashes—one for each letter in the secret word—and asks the user to guess a letter. If the user guesses a letter that appears in the word, the word is redisplayed with all instances of that letter shown in the correct positions, along with any letters guessed correctly on previous turns. If the letter does not appear in the word, the player is charged with an incorrect guess. The player keeps guessing letters until either (1) the player has correctly guessed all the letters in the word or (2) the player has made eight incorrect guesses. To separate the process of choosing a secret word from the rest of the game, define and implement an interface called randword that exports two functions: InitDictionary and ChooseRandomWord. InitDictionary has a list of words, stored into an array declared as a static global variable in the implementation. ChooseRandomWord takes no arguments and returns a word chosen at random from the internally maintained array.** **A sample run of the hangman program is shown as below**

<table>
<tr><td></td><td>

```
Welcome to Hangman!
I will guess a secret word.  On each turn, you guess
a letter.  If the letter is in the secret word, I
will show you where it appears; if not, a part of
your body gets strung up on the scaffold.  The
object is to guess the word before you are hung.
The word now looks like this: --------
You have 8 guesses left.
Guess a letter: E
That guess is correct.
The word now looks like this: ------E-
You have 8 guesses left.
Guess a letter: A
There are no A's in the word.
The word now looks like this: ------E-
You have 7 guesses left.
Guess a letter: I
There are no I's in the word.
The word now looks like this: ------E-
You have 6 guesses left.
Guess a letter: O
That guess is correct.
The word now looks like this: -O----E-
You have 6 guesses left.
Guess a letter: S
There are no S's in the word.
The word now looks like this: -O----E-
You have 5 guesses left.
Guess a letter: T
That guess is correct.
The word now looks like this: -O---TE-
You have 5 guesses left.
Guess a letter: R
That guess is correct.
The word now looks like this: -O---TER
You have 5 guesses left.
Guess a letter: N
There are no N's in the word.
The word now looks like this: -O---TER
You have 4 guesses left.
Guess a letter: P
That guess is correct.
The word now looks like this: -O-P-TER
You have 4 guesses left.
Guess a letter: C
That guess is correct.
The word now looks like this: CO-P-TER
You have 4 guesses left.
Guess a letter: M
That guess is correct.
The word now looks like this: COMP-TER
You have 4 guesses left.
Guess a letter: U
That guess is correct.
You guessed the word: COMPUTER
You win.
```

</td></tr>
<tr><td>PROGRAM:</td><td>

```
import java.util.Scanner;
interface randword{
    void InitDictionary();
    String ChooseRandomWord();
}
class working implements randword{
    static String[] words = new String[10];
    public void InitDictionary()
    {
        words[0] = "COMPUTER";
        words[1] = "JOEY";
        words[2] = "DEVELOPMENT";
```

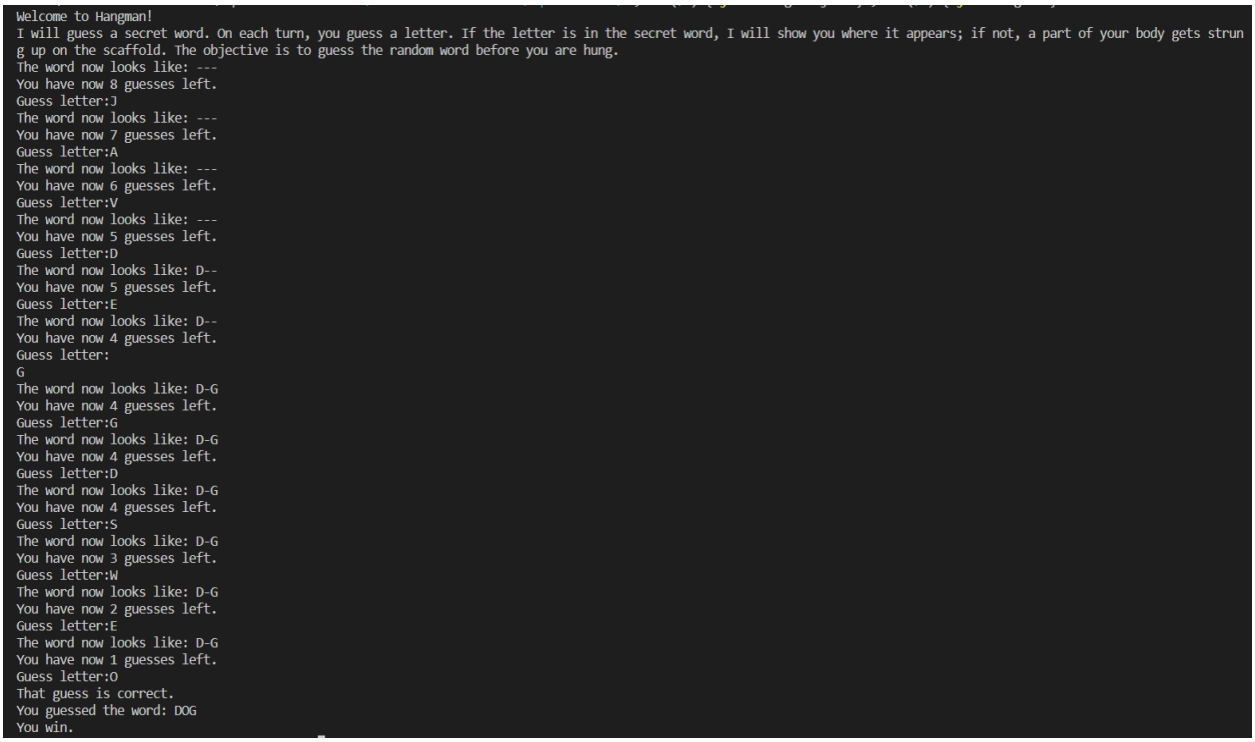
</td></tr>
</table>

```
        words[3] = "GOLDY";
        words[4] = "DOG";
        words[5] = "FLOWER";
        words[6] = "CHEMISTRY";
        words[7] = "VICTUS";
        words[8] = "LABORATORY";
        words[9] = "MORTAL";
    }
    public String ChooseRandomWord()
    {
        int randomNumber = (int)(Math.random()*9);
        return words[randomNumber];
    }
}
public class hangman {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        working obj = new working();
        obj.InitDictionary();
        String hangmanWord = obj.ChooseRandomWord();
        int guesses = 8;
        System.out.println("Welcome to Hangman!");
        System.out.println("I will guess a secret word. On each turn, you guess a letter. If the
letter is in the secret word, I will show you where it appears; if not, a part of your body gets
strung up on the scaffold. The objective is to guess the random word before you are hung.");
        char answer[] = new char[hangmanWord.length()];
        for (int i = 0; i < answer.length; i++) {
            answer[i]='-';
        }
        for(;;)
        {
            int flag = 0;
            char guess;
            System.out.print("The word now looks like: ");
            for (int j = 0; j < answer.length; j++) {
                System.out.print(answer[j]);
            }
            System.out.println("\nYou have now "+guesses+" guesses left.");
            System.out.print("Guess letter:");
            guess = sc.next().charAt(0);

            for (int j = 0; j < hangmanWord.length(); j++) {
                if (hangmanWord.charAt(j)==guess) {
```

```
                answer[j] = guess;
                flag++;
            }
        }
        if (flag==0) {
            guesses--;
        }
        int flag1 = 0;
        for (int j = 0; j < answer.length; j++) {
            if(answer[j]==hangmanWord.charAt(j))
            {
                flag1++;
            }
        }
        if (flag1==hangmanWord.length()) {
            System.out.println("That guess is correct.");
            System.out.println("You guessed the word: "+ hangmanWord);
            System.out.println("You win.");
            break;
        }
        if (guesses==0) {
            System.out.println("Sad life you have died!");
            System.out.println("Word was "+hangmanWord);
            break;
        }
    }
  }
}
```

RESULT:



Program 2

PROBLEM STATEMENT:

Design and implement an interfaces:

• A interface rankT that allows you to represent the rank of a card. The values of type rankT include the integers between 2 and 10 but should also include the constants Ace, Jack, Queen, and King.

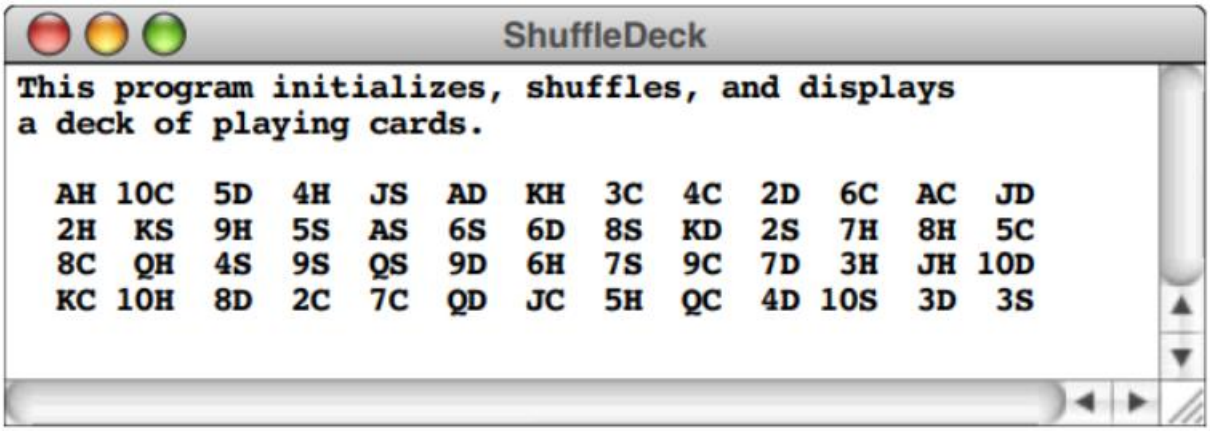
• A interface suitT consisting of the four suits: Clubs, Diamonds, Hearts, and Spades.

• A interface cardT that combines a rank and a suit.

- It has a function NewCard() that creates a card from the rank and suit values.

a) A PrintCard Class has a function CardName() that returns a string identifying the card. The result of CardName begins with a rank indicator (which is one of A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, or K), followed by a one-character suit (C, D, H, or S). Note that the result is usually a two-character string, but contains three characters if the rank is a 10.

b) Using the interfaces initialize a complete deck of 52 cards. It have method ShuffleCard() shuffles cards and then displays the shuffled values, as shown in the following sample run



PROGRAM:

```
import java.util.*;
interface rankT{ String[] rank();}
interface suitT{ String[] suit();}
class cardT implements rankT,suitT{
     public String[] rank(){
         String[] r = {"A","2","3", "4", "5", "6", "7", "8", "9", "10", "J","Q","K"};
         return r;
     }
    public String[] suit(){
        String[] s = {"C","D","H","S"};
        return s;
    }
    String[][] newCard(String[] a ,String[] b ){
         String[][] c = new String[4][13];
        for(int i =0;i<4;i++){
            for(int j =0;j<13;j++){
             c[i][j]=a[j]+b[i];
            }
        }
     return c;
    }
}
class printCard extends cardT{
    void shuffle_card(String[][] d){
        List<String> list1 = new ArrayList<>();
        for(int i =0;i<4;i++){
            for(int j =0;j<13;j++){
                list1.add(d[i][j]);
            }
        }
```

```
        Collections.shuffle(list1);
        for(int i =0;i<52;i++){
                System.out.print(list1.get(i)+"\t");
                if(i>=12 && (i+1)%13==0 && i!=51){
                    System.out.print("\n");
                }
        }
    }
}
public class card {
    public static void main(String[] args) {
        printCard p = new printCard();
        p.shuffle_card(p.newCard(p.rank(),p.suit()));
    }
}
```

RESULT:

```
4H  9C  8S  KC  QS  7D  10C  3H  QD  10S  8H  JD  AC
QH  QC  7C  5C  6C  6D  JS  7H  4C  4S  2S  6S  5D
9H  5S  9D  9S  KH  4D  JH  10D  2H  3S  2D  10H  AD
6H  AS  KD  5H  3D  3C  KS  AH  JC  2C  8C  8D  7S
```

CONCLUSION: We learned how an abstract class is used to provide a method implementation to all the subclasses or to provide a default implementation.