Title: Playfair Cipher Encryption.

Aims:

- Encrypting plaintexts using Playfair Cipher technique.

Tasks:

- Generate the key square.
- Encrypt plaintexts.

Activities:

1. Generate the key square.

```
public void setKey(String k)
    {
        String K_adjust = new String();
        boolean flag = false;
        K_adjust = K_adjust + k.charAt(0);
        for (int i = 1; i < k.length(); i++)
        {
            for (int j = 0; j < K_adjust.length(); j++)
            {
                if (k.charAt(i) == K_adjust.charAt(j))
                {
                    flag = true;
                }
            }
            if (flag == false)
                K_adjust = K_adjust + k.charAt(i);
            flag = false;
        }
        KeyWord = K_adjust;
    }

    public void KeyGen()
    {
        boolean flag = true;
        char current;
        Key = KeyWord;
        for (int i = 0; i < 26; i++)
        {
            current = (char) (i + 97);
            if (current == 'j')
                continue;
            for (int j = 0; j < KeyWord.length(); j++)
            {
                if (current == KeyWord.charAt(j))
                {
                    flag = false;
                    break;
                }
            }
            if (flag)
                Key = Key + current;
            flag = true;
        }
        System.out.println(Key);
        matrix();
    }

    private void matrix()
    {
        int counter = 0;
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                matrix_arr[i][j] = Key.charAt(counter);
                System.out.print(matrix_arr[i][j] + " ");
                counter++;
            }
            System.out.println();
        }
          }
```

2. Encrypt plaintexts.

```
package lab3_PlayFair;
import java.util.Scanner;

public class PlayFair_Cipher {
   private String KeyWord        = new String();
    private String Key            = new String();
    private char   matrix_arr[][] = new char[5][5];

    public void setKey(String k)
    {
        String K_adjust = new String();
        boolean flag = false;
        K_adjust = K_adjust + k.charAt(0);
        for (int i = 1; i < k.length(); i++)
        {
            for (int j = 0; j < K_adjust.length(); j++)
            {
                if (k.charAt(i) == K_adjust.charAt(j))
                {
                    flag = true;
                }
            }
            if (flag == false)
                K_adjust = K_adjust + k.charAt(i);
            flag = false;
        }
        KeyWord = K_adjust;
    }

    public void KeyGen()
```

```
        {
            boolean flag = true;
            char current;
            Key = KeyWord;
            for (int i = 0; i < 26; i++)
            {
                current = (char) (i + 97);
                if (current == 'j')
                    continue;
                for (int j = 0; j < KeyWord.length(); j++)
                {
                    if (current == KeyWord.charAt(j))
                    {
                        flag = false;
                        break;
                    }
                }
                if (flag)
                    Key = Key + current;
                flag = true;
            }
            System.out.println(Key);
            matrix();
        }

        private void matrix()
        {
            int counter = 0;
            for (int i = 0; i < 5; i++)
            {
                for (int j = 0; j < 5; j++)
                {
                    matrix_arr[i][j] = Key.charAt(counter);
                    System.out.print(matrix_arr[i][j] + " ");
                    counter++;
                }
                System.out.println();
            }
        }

        private String format(String old_text)
        {
            int i = 0;
            int len = 0;
            String text = new String();
            len = old_text.length();
            for (int tmp = 0; tmp < len; tmp++)
            {
                if (old_text.charAt(tmp) == 'j')
                {
                    text = text + 'i';
                }
                else
                    text = text + old_text.charAt(tmp);
            }
            len = text.length();
            for (i = 0; i < len; i = i + 2)
            {
                if (text.charAt(i + 1) == text.charAt(i))
                {
                    text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
                }
            }
            return text;
        }

        private String[] Divid2Pairs(String new_string)
        {
            String Original = format(new_string);
            int size = Original.length();
            if (size % 2 != 0)
            {
                size++;
                Original = Original + 'x';
                //System.out.println(Original);
            }
            String x[] = new String[size / 2];
            int counter = 0;
            for (int i = 0; i < size / 2; i++)
            {
                x[i] = Original.substring(counter, counter + 2);
                counter = counter + 2;
            }
            return x;
        }

        public int[] GetDiminsions(char letter)
        {
            int[] key = new int[2];
            if (letter == 'j')
                letter = 'i';
            for (int i = 0; i < 5; i++)
            {
                for (int j = 0; j < 5; j++)
                {
                    if (matrix_arr[i][j] == letter)
                    {
                        key[0] = i;
                        key[1] = j;
                        break;
                    }
                }
            }
            return key;
        }

        public String encryptMessage(String Source)
        {
            String src_arr[] = Divid2Pairs(Source);
            String Code = new String();
```

```
            char one;
            char two;
            int part1[] = new int[2];
            int part2[] = new int[2];
            for (int i = 0; i < src_arr.length; i++)
            {
                one = src_arr[i].charAt(0);
                two = src_arr[i].charAt(1);
                part1 = GetDiminsions(one);
                part2 = GetDiminsions(two);
                if (part1[0] == part2[0])
                {
                    if (part1[1] < 4)
                        part1[1]++;
                    else
                        part1[1] = 0;
                    if (part2[1] < 4)
                        part2[1]++;
                    else
                        part2[1] = 0;
                }
                else if (part1[1] == part2[1])
                {
                    if (part1[0] < 4)
                        part1[0]++;
                    else
                        part1[0] = 0;
                    if (part2[0] < 4)
                        part2[0]++;
                    else
                        part2[0] = 0;
                }
                else
                {
                    int temp = part1[1];
                    part1[1] = part2[1];
                    part2[1] = temp;
                }
                Code = Code + matrix_arr[part1[0]][part1[1]]
                        + matrix_arr[part2[0]][part2[1]];
            }
            return Code;
        }

        public static void main(String[] args)
        {
              PlayFair_Cipher x = new PlayFair_Cipher();
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter a keyword:");
            String keyword = sc.next();
            x.setKey(keyword);
            x.KeyGen();
            System.out.println("Enter word to encrypt: (Make sure length of message
    is even)");
            String key_input = sc.next();
            if (key_input.length() % 2 == 0)
            {
                System.out.println("Encryption: " + x.encryptMessage(key_input));
            }
            else
            {
                System.out.println("Message length should be even");
            }
            sc.close();
        }
    }
```