B. P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
ACADEMIC YEAR: 2018-2019 EVEN SEMESTER

Microproject/B.TECH/ECE/ SEM 6/EC 605A/2019

OBJECT ORIENTED PROGRAMMING (EC 605A)

Microproject on Synchronized Encryption

NAME Amrideep Baksi

YEAR 3rd

SEC A

UNIVERSITY ROLL 11500316137

Synchronized Encryption using Object Oriented Programming

Objective: To encode and decode a given message from user for enhanced security.

Synchronous key encryption is a type of data encryption that uses two interlocking keys such that any data that is encrypted using one key may be encoded using the other key. This method is especially useful in primitive cryptographic techniques where the sender and receiver have their own unique keys. This method ensures authenticity and privacy as well as security of data and keys.

Code:

```
import java.util.*;
class key_value{
   private int sum;
   int key_sum(String key){
      char[] key1=key.toCharArray();
      int sum,j;
      sum=j=0;
      for(int i=0;i<16;i++){
         j=(int) key1[i];
         if(key1[i]>=65 && key1[i]<90)
         j=j-64;
         else if(key1[i]>=97 && key1[i]<122)
         j=j-96;
         else if(key1[i]>=48 && key1[i]<=57)
         j=j-48;
         else
```

```
            j=j-j;


            sum=sum+j;
        }
        //System.out.println("z="+sum);
        return sum;
    }
}
class position_value{
    private int position_value;
    int pos_value(int sum,char value, int position){
        int position_value=(int) 2*value+3*position;
        //System.out.println(position_value);
        if(position_value==sum)
        return (sum+2*position);
        else
        return position_value;
    }


}
class encryption{
    private String prime_value;
    private String key;
    private char hash[]=new char[512];
    int pos_val[]=new int[16];
```

```
    void getData(String value,String key1){
        prime_value=value;
        key=key1;
    }
    String encrypt(){
        key_value k1=new key_value();
        position_value p1=new position_value();
        int z=k1.key_sum(key);
        int prime_length=prime_value.length();
        char value_char[]=prime_value.toCharArray();
        char value_key[]=key.toCharArray();
        String length_val=String.valueOf(prime_length);
        char[] length_value=length_val.toCharArray();
        Random r=new Random();
        String
alphabet="ABCDEFGHIJKLMNOPQRSTUVabcdefghijklmnopqrstuvwxyz0123456789,./
<>?;+_(*&^$%#@!):";
        int len=alphabet.length();
        for (int i=0;i<512;i++){
            hash[i]=alphabet.charAt(r.nextInt(len));
        }

        if (prime_value.length()<10){
            hash[z-2]='$';
            hash[z-1]=length_value[0];
        }
```

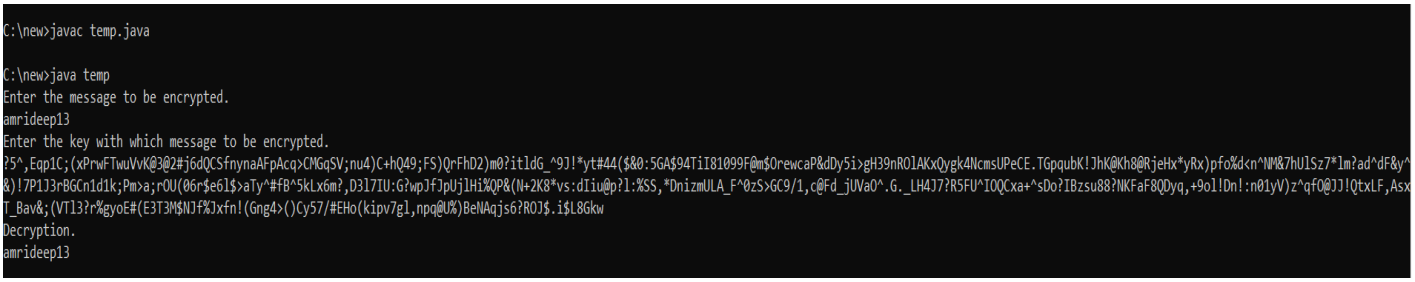
```
        else{
            hash[z-2]=length_value[0];
            hash[z-1]=length_value[1];
        }
        for (int j=0;j<16;j++){
            pos_val[j]=p1.pos_value(z,value_key[j],j);
            //System.out.println("Positon value"+pos_val[j]);
        }
        for(int j=0;j<prime_length;j++){
            hash[pos_val[j]]=value_char[j];
        }

        //hash[100]='$';
        String hash_value=new String(hash);
        return hash_value;
    }
}
class decryption{
    private String decrypt_string;
    private String hash_original;
    private String key;
    int message_length;
    void getMessage(String key1,String hash1){
        key=key1;
        hash_original=hash1;
```

```
    }
    String decrypt(){
        char key_arr[]=key.toCharArray();
        position_value pos_key1= new position_value();
        key_value k1=new key_value();
        int len_loc=k1.key_sum(key);
        char[] hash_char=hash_original.toCharArray();
        if(hash_char[len_loc-2]=='$')
        message_length=(int) hash_char[len_loc-1]-48;
        else
        message_length=(int) hash_char[len_loc-1]-38;

        int positn_value[]=new int[16];
        char[] orgmsg=new char[message_length];
        for(int j=0;j<16;j++){
            positn_value[j]=pos_key1.pos_value(len_loc,key_arr[j],j);
        }
        for(int j=0;j<message_length;j++){
            orgmsg[j]=hash_char[positn_value[j]];
        }
        String original_msg=new String(orgmsg);
        return original_msg;
        }
}
public class temp{
```

```
    public static void main(String[] args){
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter the message to be encrypted.");
        String value=scan.nextLine();
        System.out.println("Enter the key with which message to be encrypted.");
        char key[]=new char[16];
        Random r1=new Random();
        String
alphabet="ABCDEFGHIJKLMNOPQRSTUVabcdefghijklmnopqrstuvwxyz0123456789,./
<>?;+_(*&^$%#@!):";
        int len16=alphabet.length();
        for (int i=0;i<16;i++){
            key[i]=alphabet.charAt(r1.nextInt(len16));
        }
        String key1=new String(key);
        int key_length=key1.length();
        int value_length=value.length();
        if(key_length!=16 || value_length>16){
            System.out.println("Key length or value length exceeded. Must be equal to
16");

        }
        else{
        encryption ency1=new encryption();
        ency1.getData(value,key1);
        String hash_value=ency1.encrypt();
```

```
        System.out.println(hash_value);
        System.out.println("Decryption.");
        decryption decy1=new decryption();
        decy1.getMessage(key1,hash_value);
        String original_msg=decy1.decrypt();
        System.out.println(original_msg);
    }
}
}
```

Output:



Disscussion:

In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating ciphertext that can be read only if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.