DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

Institute of Engineering & Technology

Cryptography & Network Security Lab (BCSE-0071)

Submitted by:

Name: Siddhant Sharma

Uni. Roll no.: 2115990019

Section (Class roll no.): G2 (74)

Course: B. Tech (CSE)

Submitted to:

Mr. Ashish Srivastava

Assistant Professor (CEA)

Index

Experiment 1: Additive Cipher

1) Write a program to implement Additive Cipher

Objective: Implement the Additive Cipher to secure messages by shifting characters, evaluating its simplicity, and assessing its encryption strength.

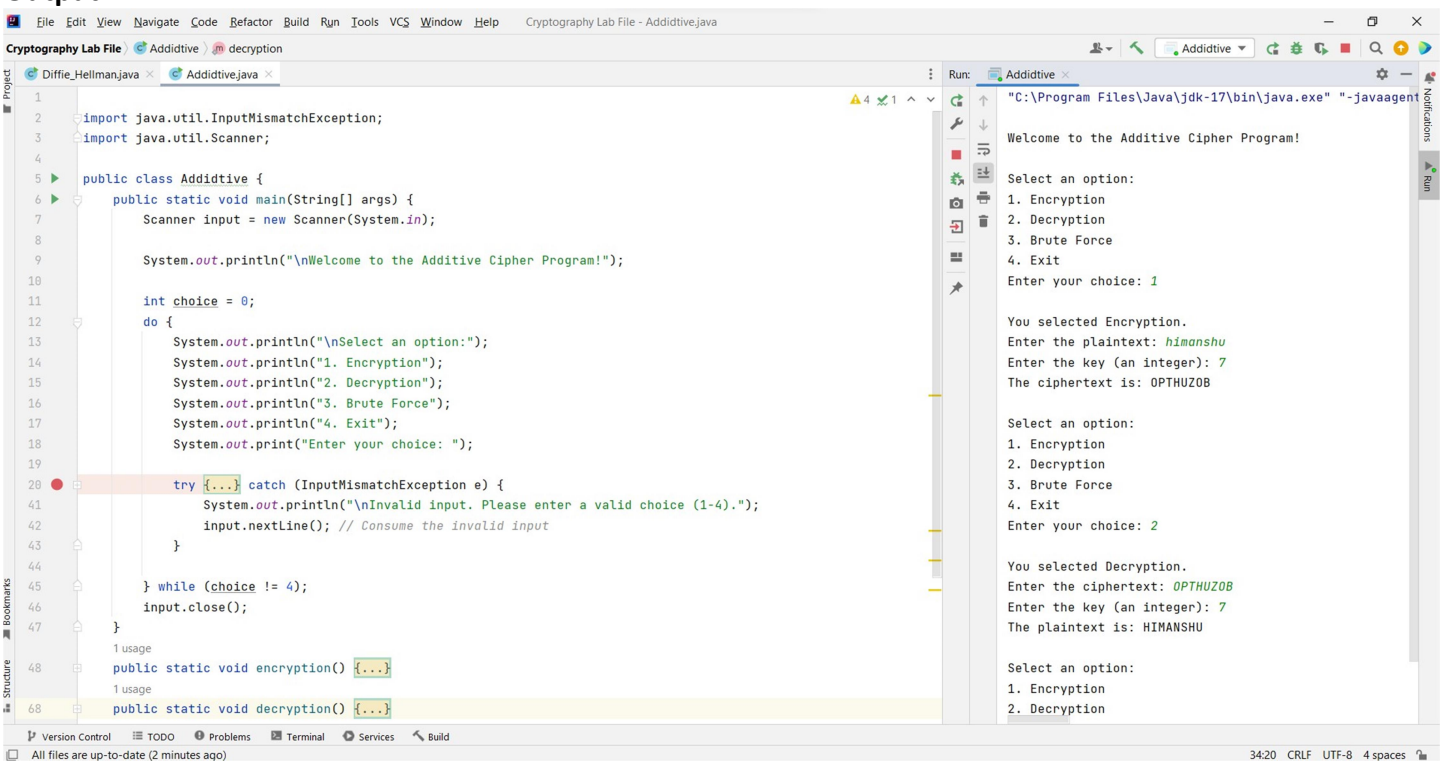
Code:

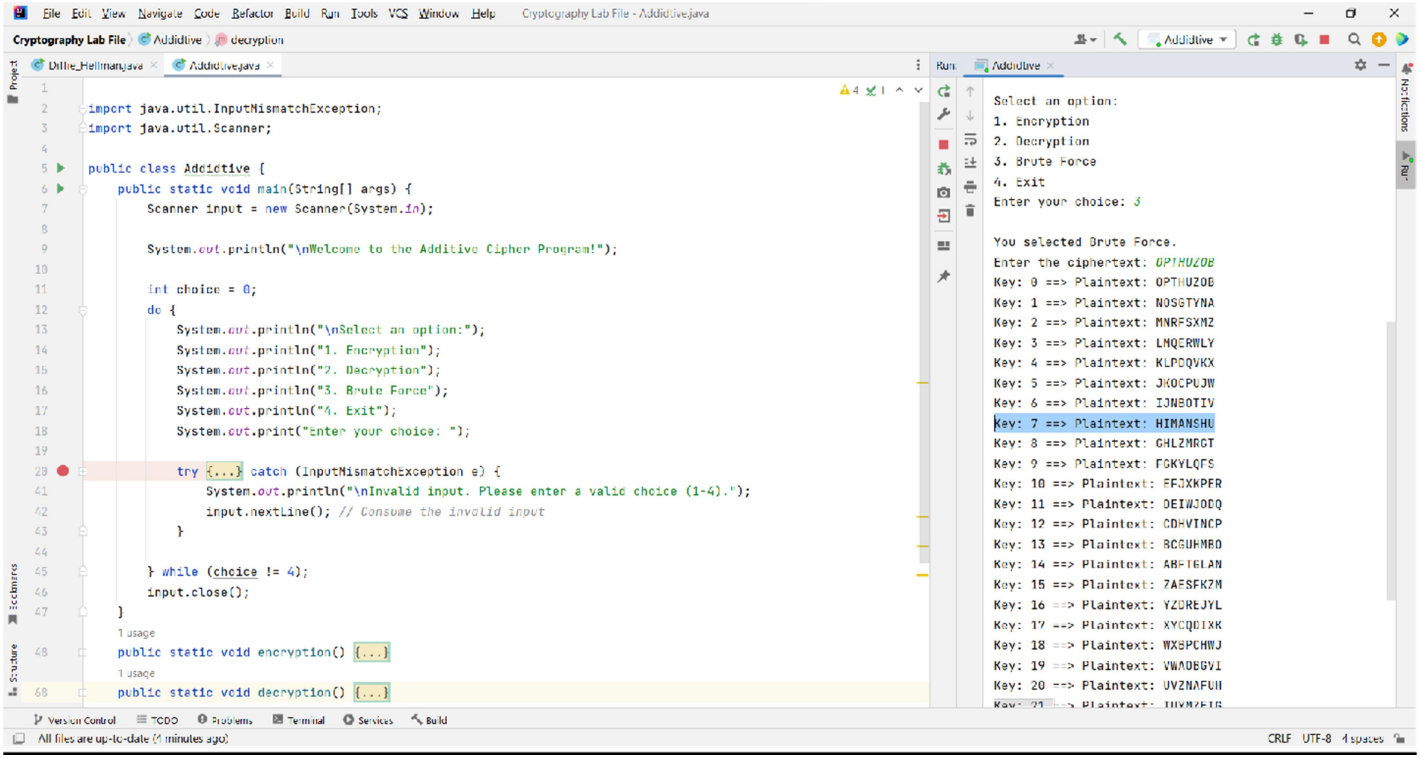
```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Addidtive {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("\nWelcome to the Additive Cipher Program!");

        int choice = 0;
        do {
            System.out.println("\nSelect an option:");
            System.out.println("1. Encryption");
            System.out.println("2. Decryption");
            System.out.println("3. Brute Force");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            try {
                choice = input.nextInt();
                input.nextLine(); // Consume the newline character
                switch (choice) {
                    case 1:
                        encryption();
                        break;
                    case 2:
                        decryption();
                        break;
                    case 3:
                        bruteForce();
                        break;
                    case 4:
                        System.out.println("\nGoodbye!");
                        break;
                    default:
                        System.out.println("\nInvalid choice. Please choose from 1-4.");
                        break;
                }
            } catch (InputMismatchException e) {
                System.out.println("\nInvalid input. Please enter a valid choice (1-4).");
                input.nextLine(); // Consume the invalid input
            }

        } while (choice != 4);
        input.close();
    }
    public static void encryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Encryption.");
```

```
        System.out.print("Enter the plaintext: ");
        String plainText = input.nextLine();

        if (!plainText.matches("[a-z ]+")) {
            System.out.println("Error: The plaintext should only contain lower letters.");
            return;
        }
        plainText = plainText.toUpperCase(); // Convert input to uppercase
        int key = getValidKey(input);
        String cipherText = "";
        for (int i = 0; i < plainText.length(); i++) {
            char ch = plainText.charAt(i);
            ch = (char) ((ch - 'A' + key) % 26 + 'A');
            cipherText += ch;
        }
        System.out.println("The ciphertext is: " + cipherText);
    }
    public static void decryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Decryption.");
        System.out.print("Enter the ciphertext: ");
        String cipherText = input.nextLine();

        // Check if the input contains non-uppercase letters
        if (!cipherText.matches("[A-Z ]+")) {
            System.out.println("Error: The ciphertext should only contain uppercase letters.");
            return;
        }
        cipherText = cipherText.toUpperCase(); // Convert input to uppercase
        int key = getValidKey(input);
        String plainText = "";
        for (int i = 0; i < cipherText.length(); i++) {
            char ch = cipherText.charAt(i);
            ch = (char) ((ch - 'A' - key + 26) % 26 + 'A');
            plainText += ch;
        }
        System.out.println("The plaintext is: " + plainText);
    }
    public static void bruteForce() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Brute Force.");
        System.out.print("Enter the ciphertext: ");
        String cipherText = input.nextLine();
        if (!cipherText.matches("[A-Z ]+")) {
            System.out.println("Error: The ciphertext should only contain uppercase letters.");
            return;
        }
        cipherText = cipherText.toUpperCase(); // Convert input to uppercase

        for (int key = 0; key < 26; key++) {
            String plainText = "";
            for (int i = 0; i < cipherText.length(); i++) {
                char ch = cipherText.charAt(i);
                ch = (char) ((ch - 'A' - key + 26) % 26 + 'A');
                plainText += ch;
            }
            System.out.println("Key: " + key + " ==> Plaintext: " + plainText);
        }
    }
    public static int getValidKey(Scanner input) {
        int key;
        do {
            System.out.print("Enter the key (an integer): ");
```

```
            while (!input.hasNextInt()) {
                System.out.println("Invalid input. Please enter an integer.");
                input.next(); // Consume the non-integer input
            }
            key = input.nextInt();
            input.nextLine(); // Consume the newline character
        } while (key < 0); // You can adjust the validation condition as needed
        return key;
    }
}
```

Output:





Experiment 2: Multiplicative Cipher

2) Write a program to implement Multiplicative Cipher

Objective: Implement the Multiplicative Cipher for message encryption, exploring its mathematical foundation, evaluating security, and understanding its cryptographic properties

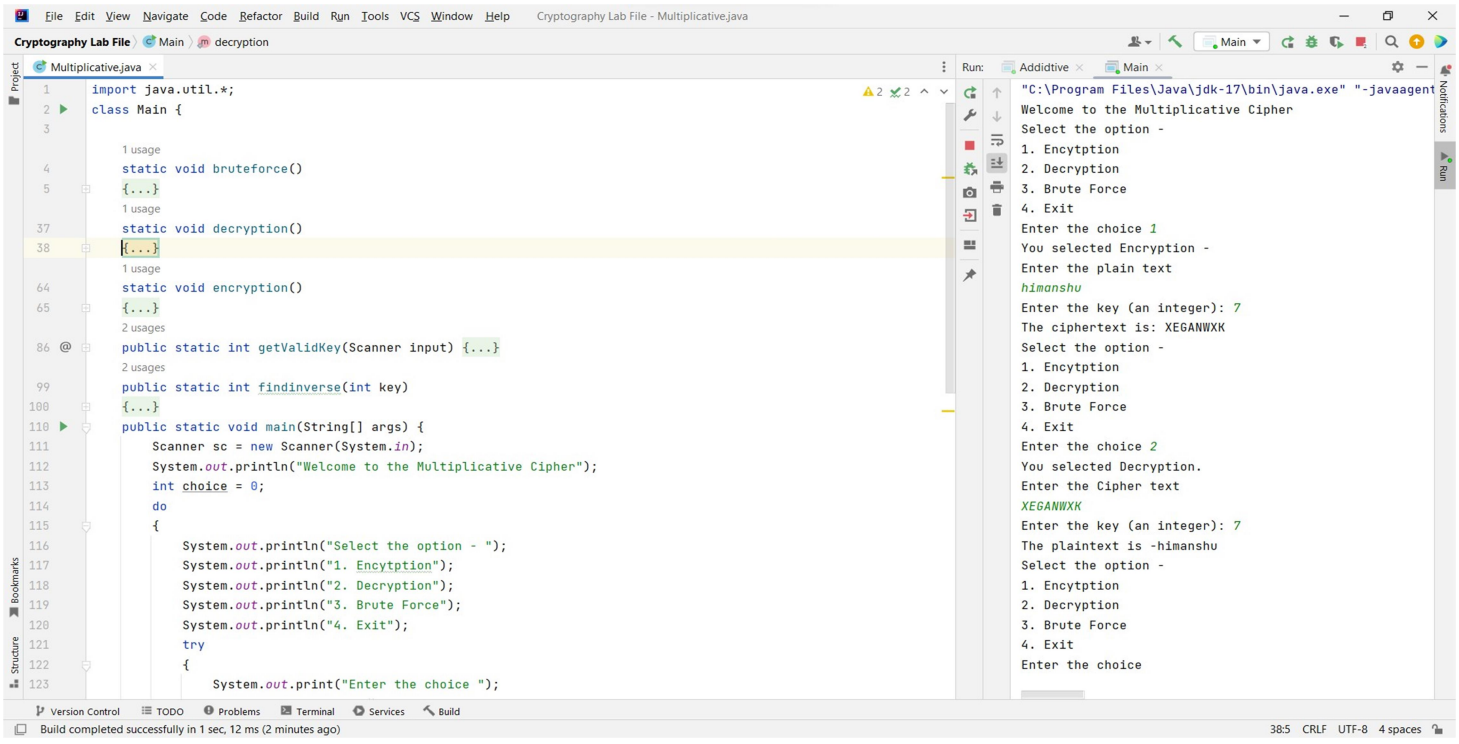
Code:

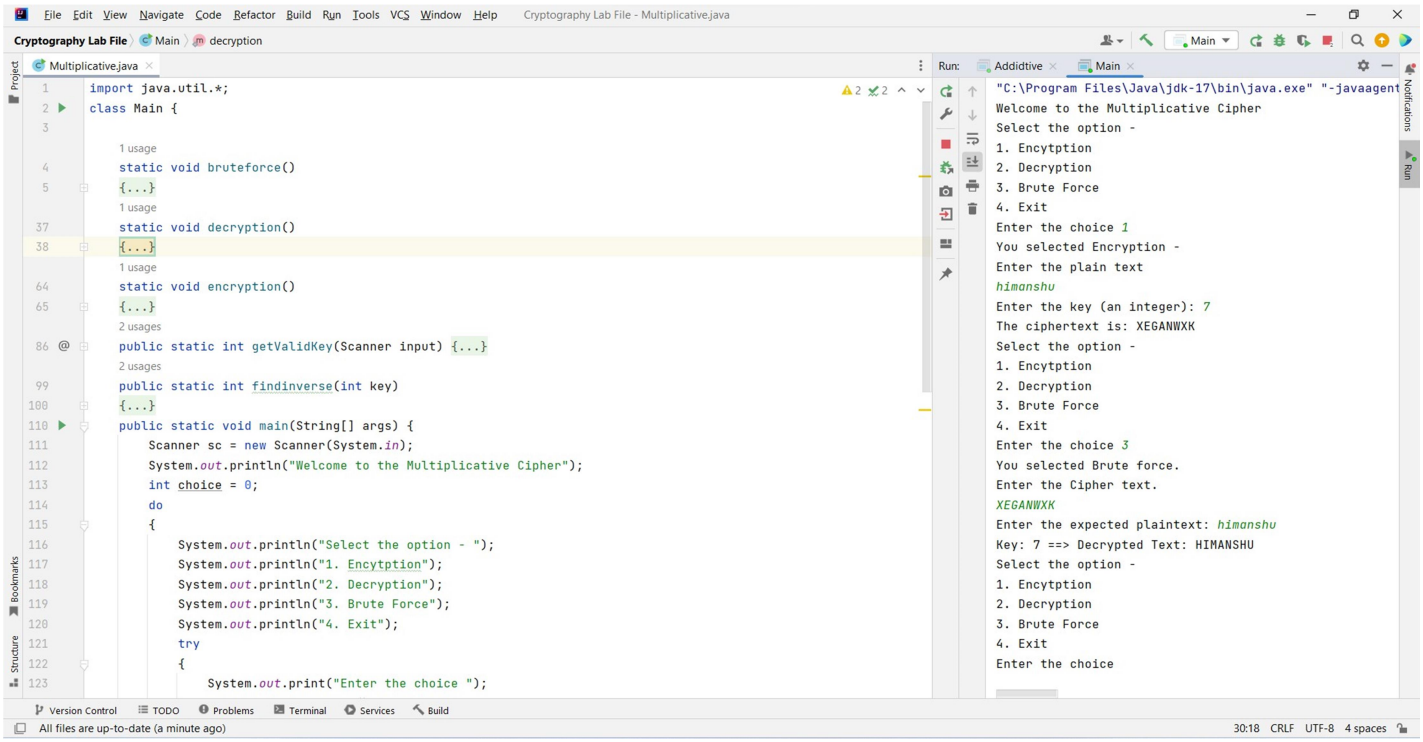
```
import java.util.*;
class Main {

    static void bruteforce()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("You selected Brute force. ");
        System.out.println("Enter the Cipher text. ");
        String ciphertext = sc.next();

        if(!ciphertext.matches("[A-Z]+"))
        {
            System.out.println("Cipher text should contains only Upper letter");
            return;
        }
        System.out.print("Enter the expected plaintext: ");
        String expectedPlainText = sc.next();

        for (int key = 1; key < 26; key++) {
            int inverseKey = findinverse(key);
            if (inverseKey != -1) {
                StringBuilder plaintext = new StringBuilder();
                for (int i = 0; i < ciphertext.length(); i++) {
                    char ch = ciphertext.charAt(i);
                    if (Character.isLetter(ch)) {
                        char base = Character.isUpperCase(ch) ? 'A' : 'a';
                        ch = (char) (base + (ch - base + 26) * inverseKey % 26);
                    }
                    plaintext.append(ch);
                }
                if (plaintext.toString().equalsIgnoreCase(expectedPlainText)) {
                    System.out.println("Key: " + key + " ==> Decrypted Text: " + plaintext);
                }
            }
        }
    }
    static void decryption()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("You selected Decryption. ");
        System.out.println("Enter the Cipher text");
        String ciphertext = sc.next();
        if(!ciphertext.matches("[A-Z]+"))
        {
            System.out.println("Cipher text should only contains Upper letters");
            return;
        }
        int key = getValidKey(sc);
        int inverseKey = findinverse(key);
        if (inverseKey == -1) {
            System.out.println("Error: The key has no multiplicative inverse in the alphabet size.");
            return;
        }
        String plaintext = "";

        for(int i=0; i<ciphertext.length(); i++)
        {
            char ch = ciphertext.charAt(i);
            ch = (char) ('A' + (ch-'A'+26)*inverseKey%26);
            plaintext+=ch;
```

```
    }
    System.out.println("The plaintext is -"+plaintext.toLowerCase());
}
static void encryption()
{
    Scanner sc = new Scanner(System.in);
    System.out.println("You selected Encryption -");
    System.out.println("Enter the plain text ");
    String plaintext = sc.next();

    if(!plaintext.matches("[a-z]+"))
    {
        System.out.println("Plain text should contains only lower Case ");
        return;
    }
    int key  =  getValidKey(sc);
    plaintext = plaintext.toUpperCase();
    StringBuilder cipherText = new StringBuilder();
    for (int i = 0; i < plaintext.length(); i++) {
        char ch = plaintext.charAt(i);
        ch = (char) ('A' + (ch - 'A') * key % 26);
        cipherText.append(ch);
    }
    System.out.println("The ciphertext is: " + cipherText.toString().toUpperCase());
}
public static int getValidKey(Scanner input) {
    int key;
    do {
        System.out.print("Enter the key (an integer): ");
        while (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.next(); // Consume the non-integer input
        }
        key = input.nextInt();
        input.nextLine(); // Consume the newline character
    } while (key < 0); // You can adjust the validation condition as needed
    return key;
}
public static int findinverse(int key)
{
    for(int i=1; i<26; i++)
    {
        if((key*i)%26==1)
        {
            return i;
        }
    }
    return -1;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Welcome to the Multiplicative Cipher");
    int choice = 0;
    do
    {
        System.out.println("Select the option - ");
        System.out.println("1. Encytption");
        System.out.println("2. Decryption");
        System.out.println("3. Brute Force");
        System.out.println("4. Exit");
        try
        {
            System.out.print("Enter the choice ");
            choice  = sc.nextInt();

            switch(choice)
            {
                case 1:
                    encryption();
                    break;
                case 2:
                    decryption();
                    break;
                case 3:
                    bruteforce();
                    break;
                case 4:
                    System.out.println("Good Bye!!!");
                    break;
```

```
                    default :
                        System.out.println("Please enter the valid choice.");
                        break;
                }
            }catch(InputMismatchException e)
            {
                System.out.println("Please Enter the valid choice");
            }
        }while(choice!=4);
        sc.close();
    }
}
```

Output:





Experiment 3: Affine Cipher

3) Write a program to implement Affine Cipher

Objective: Implement the Affine Cipher for message encryption, combining multiplication and addition, analyze security, and comprehend its cryptographic features thoroughly.

Code:

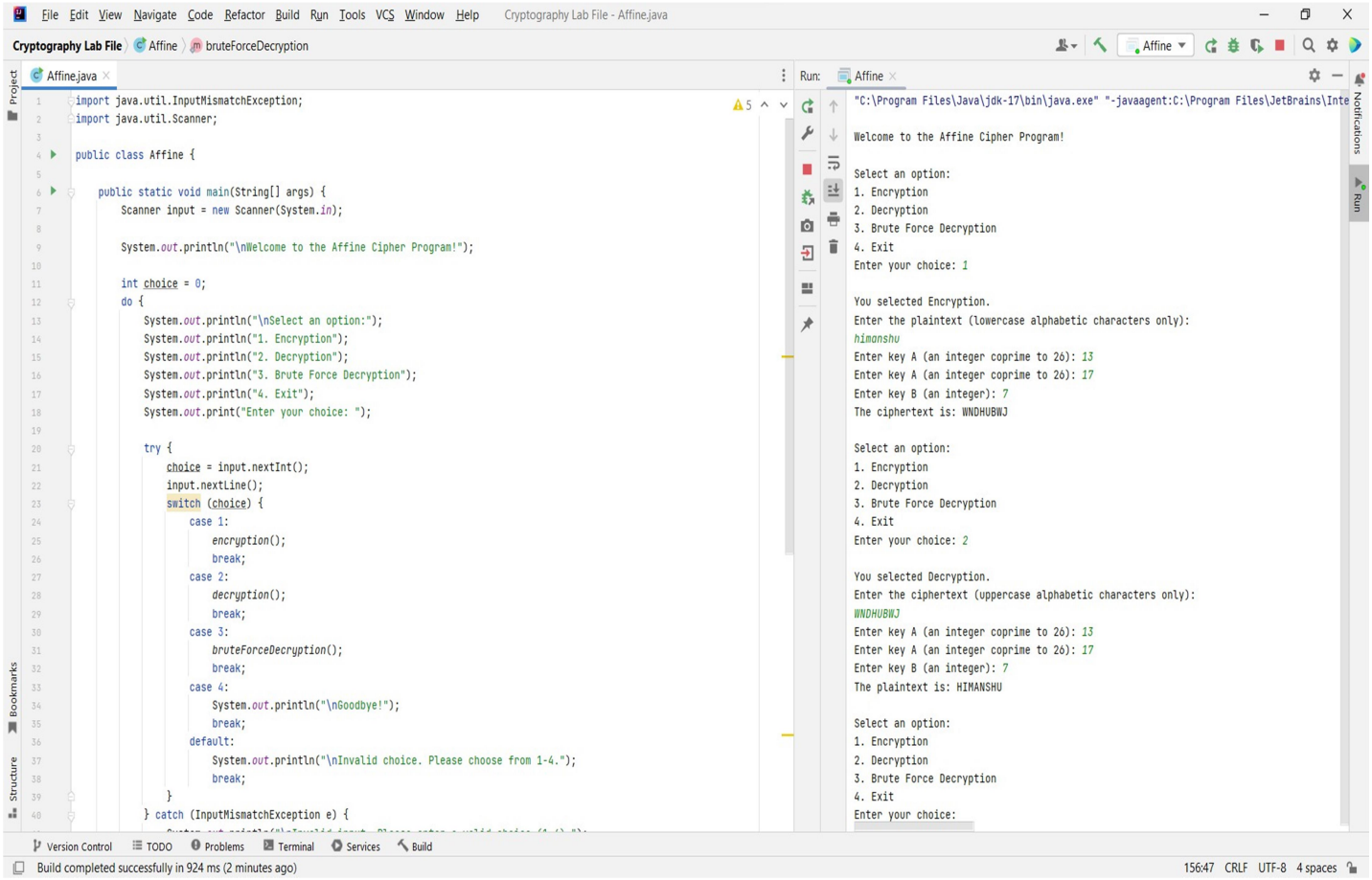
```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Affine {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("\nWelcome to the Affine Cipher Program!");

        int choice = 0;
        do {
            System.out.println("\nSelect an option:");
            System.out.println("1. Encryption");
            System.out.println("2. Decryption");
            System.out.println("3. Brute Force Decryption");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            try {
                choice = input.nextInt();
                input.nextLine();
                switch (choice) {
                    case 1:
                        encryption();
                        break;
                    case 2:
                        decryption();
                        break;
                    case 3:
                        bruteForceDecryption();
                        break;
                    case 4:
                        System.out.println("\nGoodbye!");
                        break;
                    default:
                        System.out.println("\nInvalid choice. Please choose from 1-
4.");

                        break;
                }
            } catch (InputMismatchException e) {
                System.out.println("\nInvalid input. Please enter a valid choice (1-
4).");

                input.nextLine(); // Consume the invalid input
            }

        } while (choice != 4);
        input.close();
    }
```

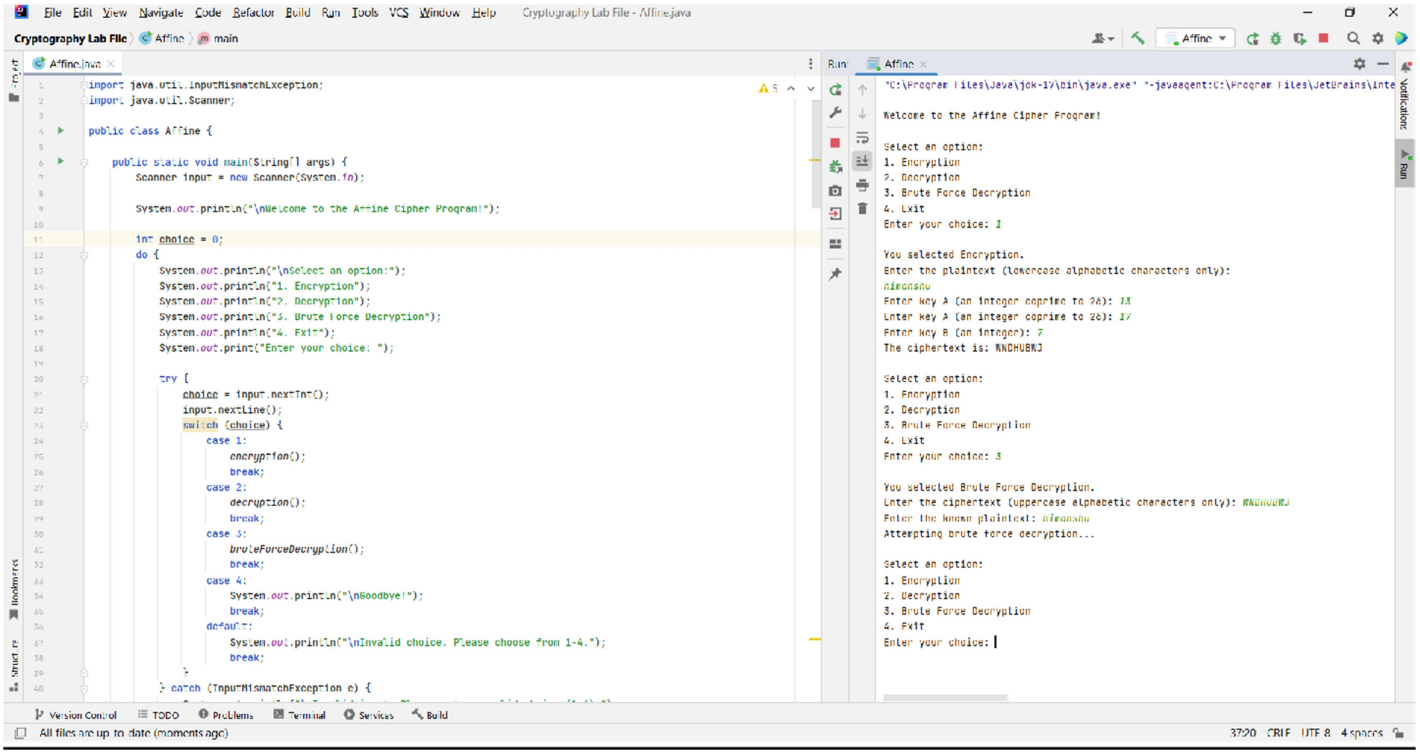
```
    public static void encryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Encryption.");

        String plainText;
        boolean validInput;

        do {
            System.out.println("Enter the plaintext (lowercase alphabetic characters
only): ");
            plainText = input.nextLine();
            validInput = plainText.matches("[a-z ]+");
            if (!validInput) {
                System.out.println("Error: The plaintext should only contain lowercase
alphabetic characters.");
            }
        } while (!validInput);

        int keyA, keyB;
        do {
            System.out.print("Enter key A (an integer coprime to 26): ");
            while (!input.hasNextInt()) {
                System.out.println("Invalid input. Please enter an integer.");
                input.next();
            }
            keyA = input.nextInt();
            input.nextLine();
        } while (!isCoprime(keyA, 26));

        System.out.print("Enter key B (an integer): ");
        while (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.next();
        }
        keyB = input.nextInt();
        input.nextLine();

        StringBuilder cipherText = new StringBuilder();
        for (int i = 0; i < plainText.length(); i++) {
            char ch = plainText.charAt(i);
            if (Character.isLetter(ch)) {
                char base = 'a';
                ch = (char) (((ch - base) * keyA + keyB) % 26 + base);
            }
            cipherText.append(ch);
        }
        System.out.println("The ciphertext is: " +
cipherText.toString().toUpperCase());

        // Close the scanner when done
        return;
    }

    public static void decryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Decryption.");
        String cipherText;

        boolean validInput;

        do {
            System.out.println("Enter the ciphertext (uppercase alphabetic characters
only): ");
```

```
            cipherText = input.nextLine();
            validInput = cipherText.matches("[A-Z ]+"); // Check if input is valid
            if (!validInput) {
                System.out.println("Error: The ciphertext should only contain
uppercase alphabetic characters.");
            }
        } while (!validInput);

        int keyA, keyB;
        do {
            System.out.print("Enter key A (an integer coprime to 26): ");
            while (!input.hasNextInt()) {
                System.out.println("Invalid input. Please enter an integer.");
                input.next();
            }
            keyA = input.nextInt();
            input.nextLine();
        } while (!isCoprime(keyA, 26));

        System.out.print("Enter key B (an integer): ");
        while (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.next();
        }
        keyB = input.nextInt();
        input.nextLine();

        int inverseKeyA = findInverse(keyA, 26);

        if (inverseKeyA == -1) {
            System.out.println("Error: Key A has no multiplicative inverse in the
alphabet size.");
            return;
        }

        StringBuilder plainText = new StringBuilder();
        for (int i = 0; i < cipherText.length(); i++) {
            char ch = cipherText.charAt(i);
            if (Character.isLetter(ch)) {
                char base = 'A';
                ch = (char) (((ch - base - keyB + 26) * inverseKeyA) % 26 + base);
            }
            plainText.append(ch);
        }
        System.out.println("The plaintext is: " + plainText);

        // Close the scanner when done
        return;
    }

    public static void bruteForceDecryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Brute Force Decryption.");

        String cipherText;
        boolean validInput;

        do {
            System.out.print("Enter the ciphertext (uppercase alphabetic characters
only): ");
            cipherText = input.nextLine();
            validInput = cipherText.matches("[A-Z ]+"); // Check if input is valid
            if (!validInput) {
```

```
                System.out.println("Error: The ciphertext should only contain
uppercase alphabetic characters.");
            }
        } while (!validInput);

        System.out.print("Enter the known plaintext: ");
        String knownPlainText = input.nextLine();

        System.out.println("Attempting brute force decryption...");

        for (int keyA = 1; keyA < 26; keyA++) {
            for (int keyB = 0; keyB < 26; keyB++) {
                int inverseKeyA = findInverse(keyA, 26);
                StringBuilder plainText = new StringBuilder();

                for (int i = 0; i < cipherText.length(); i++) {
                    char ch = cipherText.charAt(i);
                    if (Character.isLetter(ch)) {
                        char base = 'A';
                        ch = (char) (((ch - base - keyB + 26) * inverseKeyA) % 26 +
base);
                    }
                    plainText.append(ch);
                }
                if (plainText.toString().equals(knownPlainText)) {
                    System.out.println("Key A: " + keyA + ", Key B: " + keyB + " - " +
knownPlainText);
                }
            }
        }
        return;
    }
    public static boolean isCoprime(int a, int b) {
        // Check if two numbers are coprime (have gcd = 1)
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a == 1;
    }
    public static int findInverse(int a, int m) {
        for (int x = 1; x < m; x++) {
            if ((a * x) % m == 1) {
                return x;
            }
        }
        return -1; // No modular inverse exists
    }
}
```

Output:


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Cryptography Lab File - Affine.java
Cryptography Lab File
Affine
bruteForceDecryption
Affine.java
Run:
Affine
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
Welcome to the Affine Cipher Program!
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice: 1
You selected Encryption.
Enter the plaintext (lowercase alphabetic characters only):
himanshu
Enter key A (an integer coprime to 26): 13
Enter key A (an integer coprime to 26): 17
Enter key B (an integer): 7
The ciphertext is: WNDHUBWJ
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice: 2
You selected Decryption.
Enter the ciphertext (uppercase alphabetic characters only):
WNDHUBWJ
Enter key A (an integer coprime to 26): 13
Enter key A (an integer coprime to 26): 17
Enter key B (an integer): 7
The plaintext is: HIMANSHU
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice:
Version Control
TODO
Problems
Terminal
Services
Build
Build completed successfully in 924 ms (2 minutes ago)
156:47
CRLF
UTF-8
4 spaces


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Cryptography Lab File - Affine.java
Cryptography Lab File
Affine
main
Affine.java
Run:
Affine
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
Welcome to the Affine Cipher Program!
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice: 1
You selected Encryption.
Enter the plaintext (lowercase alphabetic characters only):
himanshu
Enter key A (an integer coprime to 26): 13
Enter key A (an integer coprime to 26): 17
Enter key B (an integer): 7
The ciphertext is: WNDHUBWJ
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice: 3
You selected Brute Force Decryption.
Enter the ciphertext (uppercase alphabetic characters only): WNDHUBWJ
Enter the known plaintext: himanshu
Attempting brute force decryption...
Select an option:
1. Encryption
2. Decryption
3. Brute Force Decryption
4. Exit
Enter your choice:
Version Control
TODO
Problems
Terminal
Services
Build
All files are up to date (moments ago)
37:20
CRLF
UTF-8
4 spaces

Experiment 4: Autokey Cipher

4) Write a program in to implement Autokey Cipher

Objective: Implement the Autokey Cipher for secure communication, utilizing a key derived from plaintext, and evaluate its encryption strength and efficiency

Code:

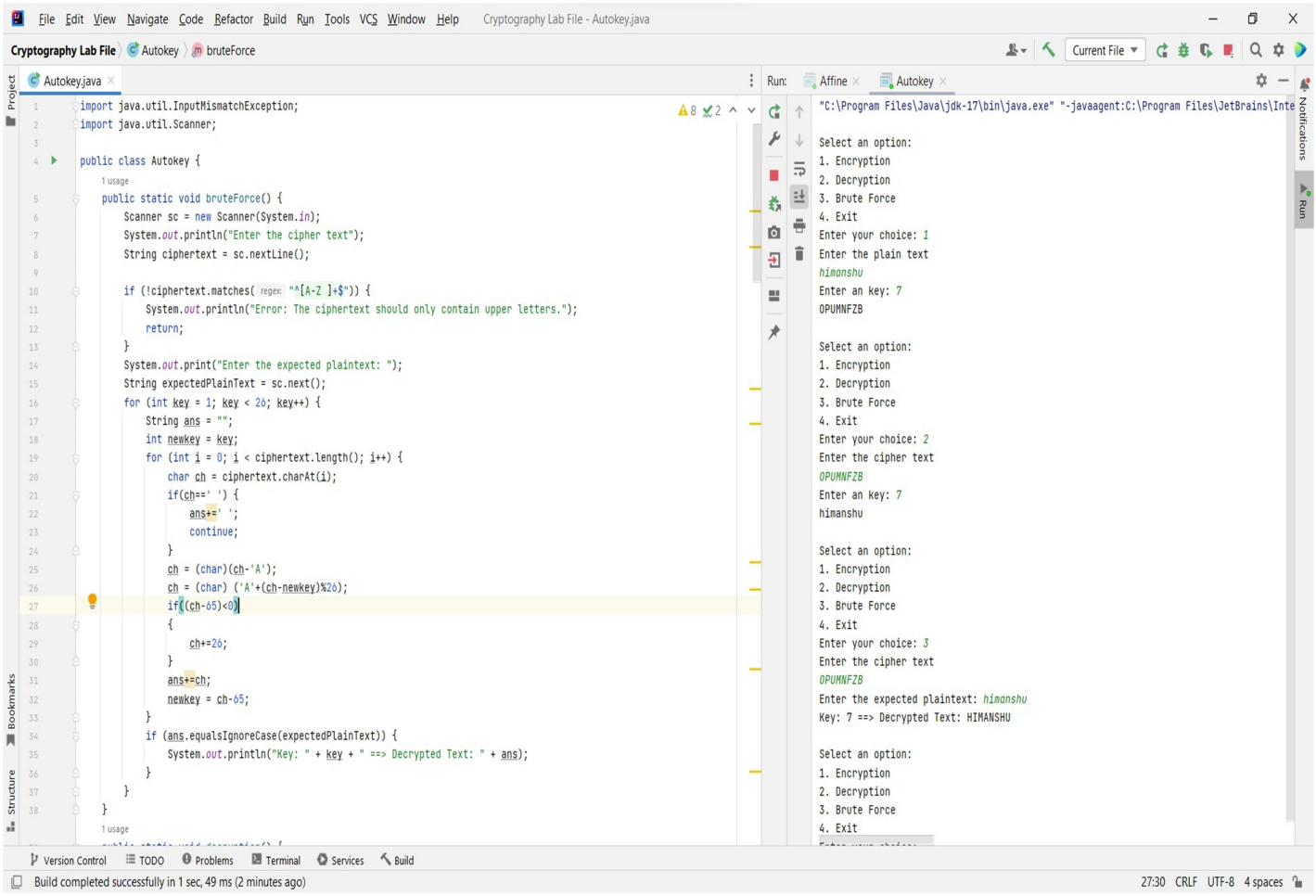
```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Autokey {
    public static void bruteForce() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the cipher text");
        String ciphertext = sc.nextLine();

        if (!ciphertext.matches("^[A-Z ]+$")) {
            System.out.println("Error: The ciphertext should only contain upper
letters.");
            return;
        }
        System.out.print("Enter the expected plaintext: ");
        String expectedPlainText = sc.next();
        for (int key = 1; key < 26; key++) {
            String ans = "";
            int newkey = key;
            for (int i = 0; i < ciphertext.length(); i++) {
                char ch = ciphertext.charAt(i);
                if(ch==' ') {
                    ans+=' ';
                    continue;
                }
                ch = (char)(ch-'A');
                ch = (char) ('A'+(ch-newkey)%26);
                if((ch-65)<0)
                {
                    ch+=26;
                }
                ans+=ch;
                newkey = ch-65;
            }
            if (ans.equalsIgnoreCase(expectedPlainText)) {
                System.out.println("Key: " + key + " ==> Decrypted Text: " + ans);
            }
        }
    }
    public static void decryption() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the cipher text");
        String ciphertext = sc.nextLine();

        if (!ciphertext.matches("[A-Z ]+")) {
            System.out.println("Error: The ciphertext should only contain upper
letters.");
            return;
        }
```

```
        int key = 0;
        boolean isValidInput = false;

        while (!isValidInput) {
            try {
                System.out.print("Enter an key: ");
                key = sc.nextInt();
                isValidInput = true;
            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please enter a valid integer.");
                sc.nextLine();
            }
        }
        String ans = "";
        for (int i = 0; i < ciphertext.length(); i++) {
            char ch = ciphertext.charAt(i);
            if(ch==' ') {
                ans+=' ';
                continue;
            }
            ch = (char)(ch-'A');
            ch = (char) ('A'+(ch-key)%26);
            if((ch-65)<0)
            {
                ch+=26;
            }
            ans+=ch;
            key = ch-65;
        }
        System.out.println(ans.toLowerCase());
    }
    public static void encryption()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the plain text");
        String plaintext = sc.nextLine();

        if (!plaintext.matches("[a-z ]+")) {
            System.out.println("Error: The plaintext should only contain lower
letters.");
            return;
        }
        int key = 0;
        boolean isValidInput = false;

        while (!isValidInput) {
            try {
                System.out.print("Enter an key: ");
                key = sc.nextInt();
                isValidInput = true;
            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please enter a valid integer.");
                sc.nextLine();

            }
        }
        String ans = "";
        for (int i = 0; i <plaintext.length(); i++) {
            int ch = plaintext.codePointAt(i);
            if(ch==' ')
            {
                ans+=' ';
                continue;
```

```
            }
            int newkey = plaintext.codePointAt(i);
            ch=ch-97;
            ch=((ch+key)%26);
            char newChar = (char) (ch+97 );
            ans+=newChar;
            key=newkey-97;
        }
        System.out.println(ans.toUpperCase());

    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int choice = 0;
        do {
            System.out.println("\nSelect an option:");
            System.out.println("1. Encryption");
            System.out.println("2. Decryption");
            System.out.println("3. Brute Force");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            try {
                choice = input.nextInt();
                input.nextLine(); // Consume the newline character
                switch (choice) {
                    case 1:
                        encryption();
                        break;
                    case 2:
                        decryption();
                        break;
                    case 3:
                        bruteForce();
                        break;

                    case 4:
                        System.out.println("\nGoodbye!");
                        break;
                    default:
                        System.out.println("\nInvalid choice. Please choose from 1-
4.");

                        break;
                }
            } catch (InputMismatchException e) {
                System.out.println("\nInvalid input. Please enter a valid choice (1-
4).");
                input.nextLine(); // Consume the invalid input
            }
        } while (choice != 4);
        input.close();
    }
    public static int getValidKey(Scanner input) {
        int key;
        do {
            System.out.print("Enter the key (an integer): ");
            while (!input.hasNextInt()) {
                System.out.println("Invalid input. Please enter an integer.");
                input.next(); // Consume the non-integer input
            }
            key = input.nextInt();
            input.nextLine(); // Consume the newline character
        } while (key < 0); // You can adjust the validation condition as needed
```

```
        return key;
    }
}
```

Output:



Experiment 5: Playfair Cipher

5) Write a program to implement Playfair Cipher to encrypt & decrypt the given message where the key matrix can be formed by using a given keyword.

Objective: Implement the Playfair Cipher for secure text encryption, understanding its grid-based algorithm, evaluating its effectiveness, and ensuring cryptographic robustness.

Code:

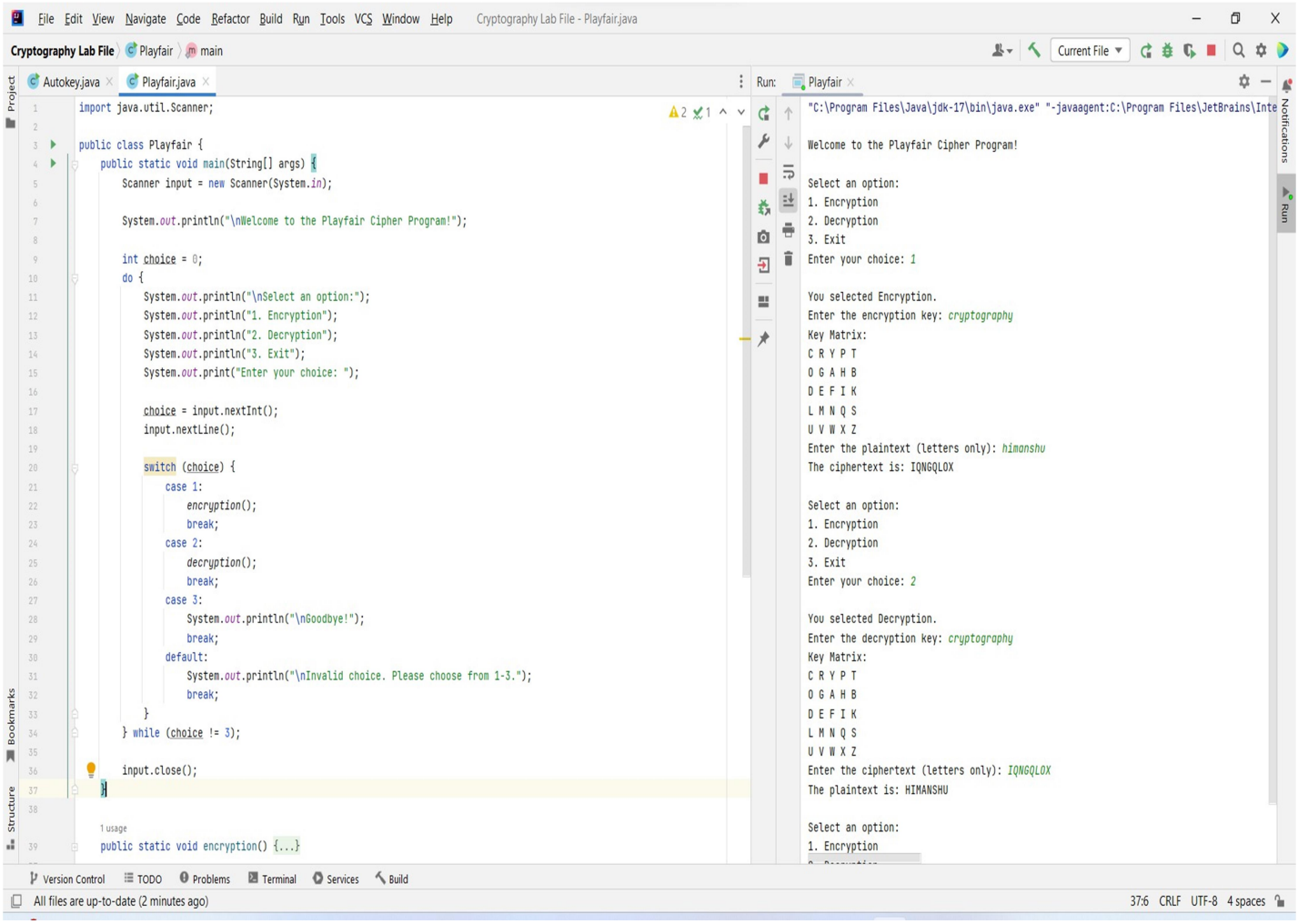
```
import java.util.Scanner;

public class Playfair {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("\nWelcome to the Playfair Cipher Program!");

        int choice = 0;
        do {
            System.out.println("\nSelect an option:");
            System.out.println("1. Encryption");
            System.out.println("2. Decryption");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");

            choice = input.nextInt();
            input.nextLine();

            switch (choice) {
                case 1:
                    encryption();
                    break;
                case 2:
                    decryption();
                    break;
                case 3:
                    System.out.println("\nGoodbye!");
                    break;
                default:
                    System.out.println("\nInvalid choice. Please choose from 1-3.");
                    break;
            }
        } while (choice != 3);

        input.close();
    }

    public static void encryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Encryption.");

        System.out.print("Enter the encryption key: ");
        String key = input.nextLine().toUpperCase(); // Convert to uppercase
        char[][] keyMatrix = createKeyMatrix(key);

        // Display the key matrix
        System.out.println("Key Matrix:");
```

```
        displayMatrix(keyMatrix);

        System.out.print("Enter the plaintext (letters only): ");
        String plainText = input.nextLine().toUpperCase().replaceAll("[^A-Z]", ""); //
Remove non-uppercase letters

        String cipherText = playfairEncrypt(plainText, keyMatrix);
        System.out.println("The ciphertext is: " + cipherText);
    }

    public static void decryption() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nYou selected Decryption.");

        System.out.print("Enter the decryption key: ");
        String key = input.nextLine().toUpperCase(); // Convert to uppercase
        char[][] keyMatrix = createKeyMatrix(key);

        // Display the key matrix
        System.out.println("Key Matrix:");
        displayMatrix(keyMatrix);

        System.out.print("Enter the ciphertext (letters only): ");
        String cipherText = input.nextLine().toUpperCase().replaceAll("[^A-Z]", "");
// Remove non-uppercase letters

        String plainText = playfairDecrypt(cipherText, keyMatrix);
        System.out.println("The plaintext is: " + plainText);
    }

    public static void displayMatrix(char[][] matrix) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static char[][] createKeyMatrix(String key) {
        char[][] matrix = new char[5][5];
        String keyWithoutDuplicates = removeDuplicates(key +
"ABCDEFGHIKLMNOPQRSTUVWXYZ");

        int row = 0, col = 0;
        for (int i = 0; i < keyWithoutDuplicates.length(); i++) {
            char ch = keyWithoutDuplicates.charAt(i);
            matrix[row][col] = ch;
            col++;
            if (col == 5) {
                row++;
                col = 0;
            }
        }

        return matrix;
    }

    public static String removeDuplicates(String str) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (result.indexOf(String.valueOf(ch)) == -1) {
```

```
                result.append(ch);
            }
        }
        return result.toString();
    }

    public static String playfairEncrypt(String plainText, char[][] keyMatrix) {
        StringBuilder cipherText = new StringBuilder();

        for (int i = 0; i < plainText.length(); i += 2) {
            char firstChar = plainText.charAt(i);
            char secondChar = (i + 1 < plainText.length()) ? plainText.charAt(i + 1) :
'X';

            int[] firstCharPos = findCharPosition(firstChar, keyMatrix);
            int[] secondCharPos = findCharPosition(secondChar, keyMatrix);

            int row1 = firstCharPos[0];
            int col1 = firstCharPos[1];
            int row2 = secondCharPos[0];
            int col2 = secondCharPos[1];

            if (row1 == row2) {
                col1 = (col1 + 1) % 5;
                col2 = (col2 + 1) % 5;
            } else if (col1 == col2) {
                row1 = (row1 + 1) % 5;
                row2 = (row2 + 1) % 5;
            } else {
                int temp = col1;
                col1 = col2;
                col2 = temp;
            }

            cipherText.append(keyMatrix[row1][col1]);
            cipherText.append(keyMatrix[row2][col2]);
        }

        return cipherText.toString();
    }

    public static String playfairDecrypt(String cipherText, char[][] keyMatrix) {
        StringBuilder plainText = new StringBuilder();

        for (int i = 0; i < cipherText.length(); i += 2) {
            char firstChar = cipherText.charAt(i);
            char secondChar = cipherText.charAt(i + 1);

            int[] firstCharPos = findCharPosition(firstChar, keyMatrix);
            int[] secondCharPos = findCharPosition(secondChar, keyMatrix);

            int row1 = firstCharPos[0];
            int col1 = firstCharPos[1];
            int row2 = secondCharPos[0];
            int col2 = secondCharPos[1];
            if (row1 == row2) {
                col1 = (col1 - 1 + 5) % 5;
                col2 = (col2 - 1 + 5) % 5;
            } else if (col1 == col2) {
                row1 = (row1 - 1 + 5) % 5;
                row2 = (row2 - 1 + 5) % 5;
            } else {
                int temp = col1;
```

```
                col1 = col2;
                col2 = temp;
            }
            plainText.append(keyMatrix[row1][col1]);
            plainText.append(keyMatrix[row2][col2]);
        }
        return plainText.toString();
    }
    public static int[] findCharPosition(char ch, char[][] keyMatrix) {
        int[] position = new int[2];
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (keyMatrix[i][j] == ch) {
                    position[0] = i;
                    position[1] = j;
                    return position;
                }
            }
        }
        return position;
    }
}
```

Output:



Experiment 6: Diffie-Hellman Cipher

Write a program to implement Diffie-Hellman key exchange Algorithm to exchange the symmetric key and show the encryption & decryption.

Objective: Implement the Diffie-Hellman Cipher for secure key exchange, exploring its mathematical principles, assessing security, and ensuring cryptographic reliability.

Code:

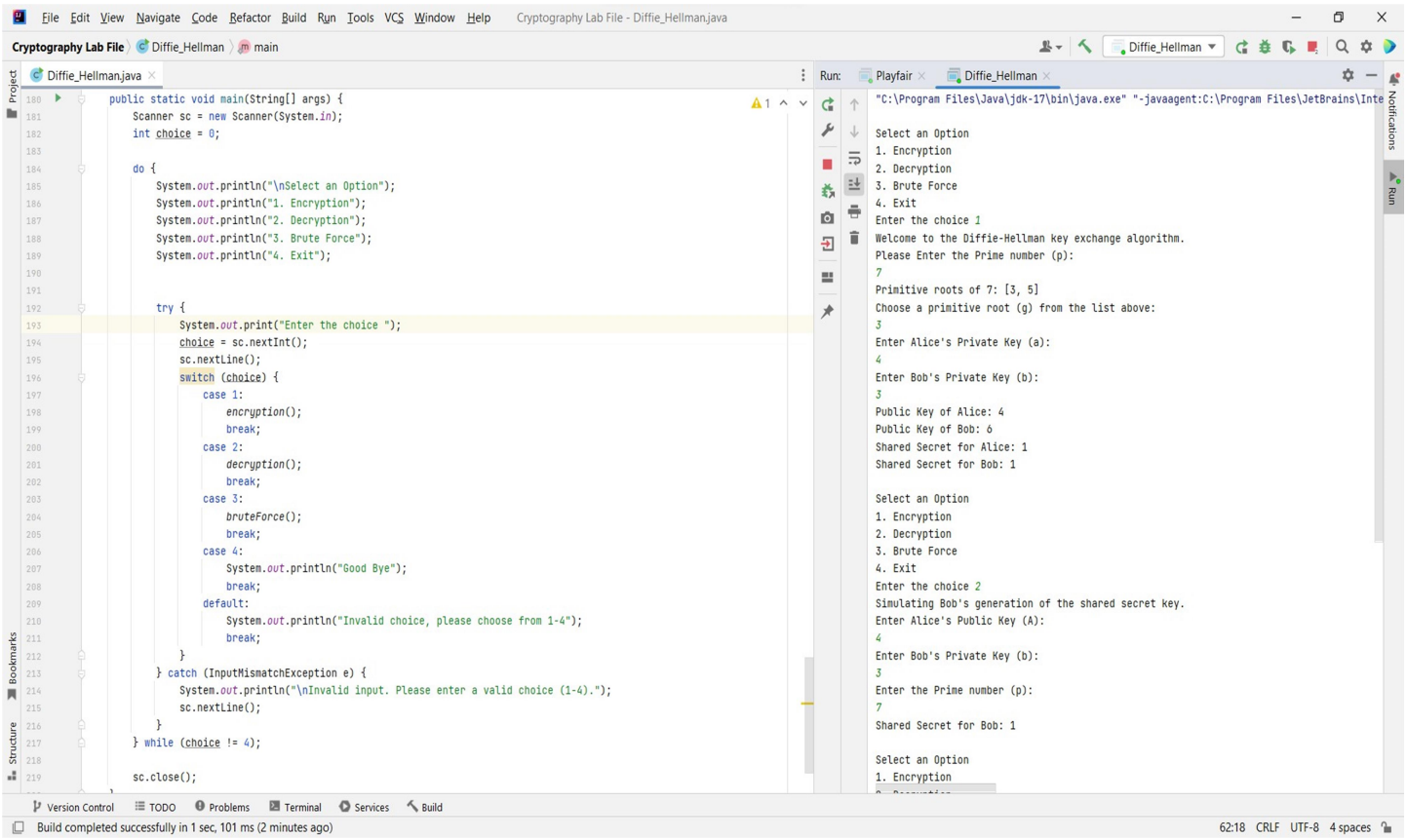
```
import java.util.*;

class Diffie_Hellman  {

    static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }

        if (num <= 3) {
            return true;
        }

        if (num % 2 == 0 || num % 3 == 0) {
            return false;
        }


        for (int i = 5; i * i <= num; i += 6) {
            if (num % i == 0 || num % (i + 2) == 0) {
                return false;
            }
        }

        return true;
    }
    public static ArrayList<Integer> findPrimitiveRoots(int p) {
        ArrayList<Integer> primitiveRoots = new ArrayList<>();
        if (isPrime(p)) {
            for (int i = 2; i < p; i++) {
                if (isPrimitiveRoot(i, p)) {
                    primitiveRoots.add(i);
                }
            }
        }
        return primitiveRoots;
    }


    public static boolean isPrimitiveRoot(int candidate, int p) {
        if (gcd(candidate, p) != 1) {
            return false;
        }

        ArrayList<Integer> powers = new ArrayList<>();
        int x = 1;
        powers.add(x);

        for (int i = 1; i < p - 1; i++) {
```

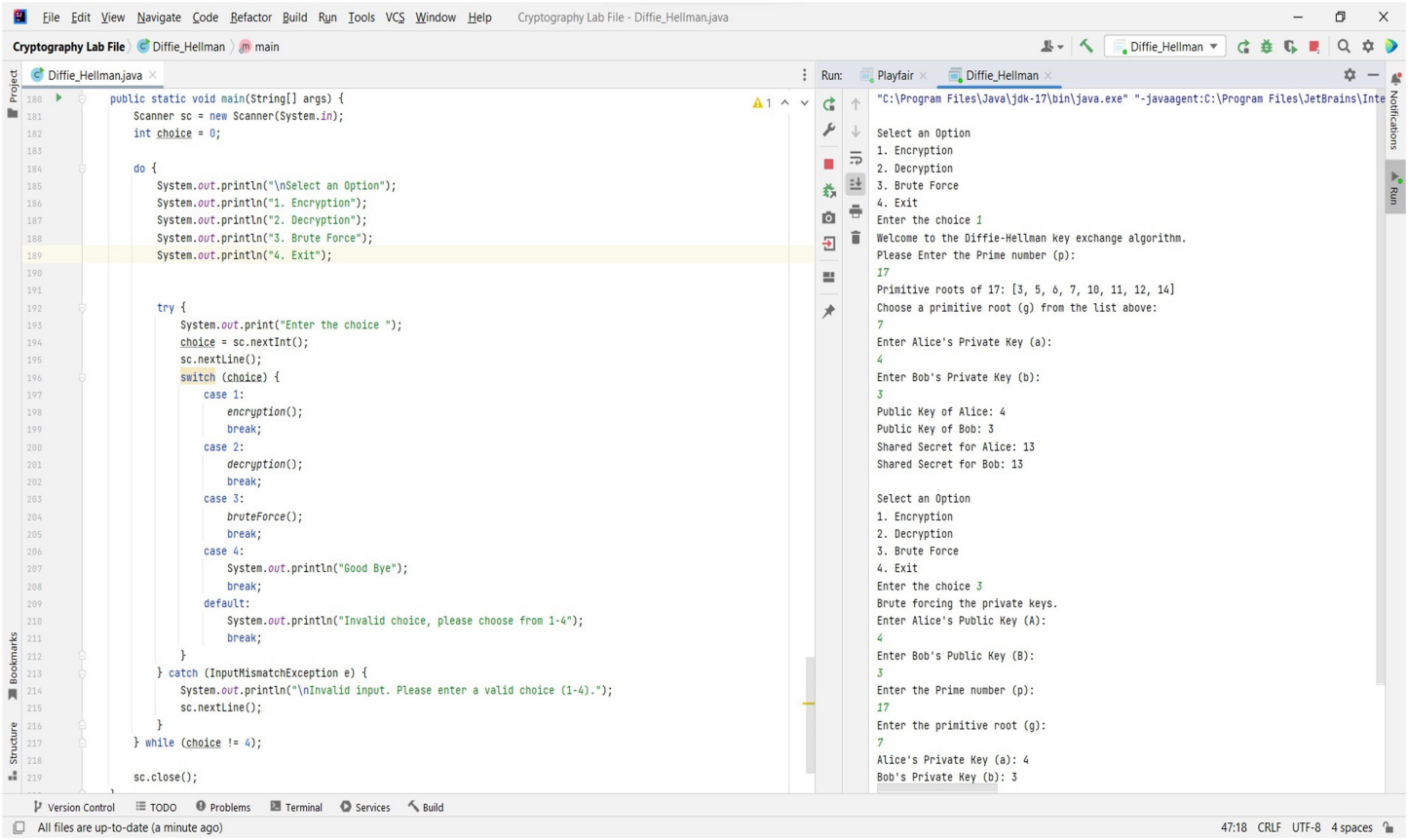
```
            x = (x * candidate) % p;
            if (powers.contains(x)) {
                return false;
            }
            powers.add(x);
        }

        return true;
    }

    public static int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }

    static void encryption() {
        System.out.println("Welcome to the Diffie-Hellman key exchange algorithm.");
        Scanner sc = new Scanner(System.in);
        System.out.println("Please Enter the Prime number (p):");
        int p = sc.nextInt();

        if (!isPrime(p)) {
            System.out.println("Please enter a prime number.");
            return;
        }

        ArrayList<Integer> primitiveRoots = findPrimitiveRoots(p);

        if (primitiveRoots.isEmpty()) {
            System.out.println("Sorry, no primitive roots found for " + p);
            return;
        }

        System.out.println("Primitive roots of " + p + ": " + primitiveRoots);

        System.out.println("Choose a primitive root (g) from the list above:");
        int g = sc.nextInt();

        if (!primitiveRoots.contains(g)) {
            System.out.println(g + " is not a valid primitive root of " + p);
            return;
        }

        System.out.println("Enter Alice's Private Key (a):");
        int a = sc.nextInt();

        System.out.println("Enter Bob's Private Key (b):");
        int b = sc.nextInt();


        int A = (int) (Math.pow(g, a) % p);
        int B = (int) (Math.pow(g, b) % p);

        System.out.println("Public Key of Alice: " + A);
        System.out.println("Public Key of Bob: " + B);


        int secretA = (int) (Math.pow(B, a) % p);
        int secretB = (int) (Math.pow(A, b) % p);

        System.out.println("Shared Secret for Alice: " + secretA);
```

```
        System.out.println("Shared Secret for Bob: " + secretB);


    }


    static void decryption() {
        System.out.println("Simulating Bob's generation of the shared secret key.");
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Alice's Public Key (A):");
        int A = sc.nextInt();
        System.out.println("Enter Bob's Private Key (b):");
        int b = sc.nextInt();
        System.out.println("Enter the Prime number (p):");
        int p = sc.nextInt();

        int secretB = (int) (Math.pow(A, b) % p);

        System.out.println("Shared Secret for Bob: " + secretB);
    }

    static void bruteForce() {
        System.out.println("Brute forcing the private keys.");
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Alice's Public Key (A):");
        int A = sc.nextInt();
        System.out.println("Enter Bob's Public Key (B):");
        int B = sc.nextInt();
        System.out.println("Enter the Prime number (p):");
        int p = sc.nextInt();
        System.out.println("Enter the primitive root (g):");
        int g = sc.nextInt();

        int a = -1;
        int b = -1;

        // Brute force to find Alice's private key
        for (int i = 0; i < p; i++) {
            if ((int) (Math.pow(g, i) % p) == A) {
                a = i;
                break;
            }
        }

        // Brute force to find Bob's private key
        for (int i = 0; i < p; i++) {
            if ((int) (Math.pow(g, i) % p) == B) {
                b = i;
                break;
            }
        }

        if (a != -1) {
            System.out.println("Alice's Private Key (a): " + a);
        } else {
            System.out.println("Could not find Alice's private key.");
        }

        if (b != -1) {
            System.out.println("Bob's Private Key (b): " + b);
        } else {
            System.out.println("Could not find Bob's private key.");
        }
```

```
    }



    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice = 0;

        do {
            System.out.println("\nSelect an Option");
            System.out.println("1. Encryption");
            System.out.println("2. Decryption");
            System.out.println("3. Brute Force");
            System.out.println("4. Exit");


            try {
                System.out.print("Enter the choice ");
                choice = sc.nextInt();
                sc.nextLine();
                switch (choice) {
                    case 1:
                        encryption();
                        break;
                    case 2:
                        decryption();
                        break;
                    case 3:
                        bruteForce();
                        break;
                    case 4:
                        System.out.println("Good Bye");
                        break;
                    default:
                        System.out.println("Invalid choice, please choose from 1-4");
                        break;
                }
            } catch (InputMismatchException e) {
                System.out.println("\nInvalid input. Please enter a valid choice (14).");
                sc.nextLine();
            }
        } while (choice != 4);

        sc.close();
    }
}
```

Output:


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Cryptography Lab File - Diffie_Hellman.java
Cryptography Lab File > Diffie_Hellman > main
Diffie_Hellman.java
Run: Playfair Diffie_Hellman
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
Select an Option
1. Encryption
2. Decryption
3. Brute Force
4. Exit
Enter the choice 1
Welcome to the Diffie-Hellman key exchange algorithm.
Please Enter the Prime number (p):
7
Primitive roots of 7: [3, 5]
Choose a primitive root (g) from the list above:
3
Enter Alice's Private Key (a):
4
Enter Bob's Private Key (b):
3
Public Key of Alice: 4
Public Key of Bob: 6
Shared Secret for Alice: 1
Shared Secret for Bob: 1
Select an Option
1. Encryption
2. Decryption
3. Brute Force
4. Exit
Enter the choice 2
Simulating Bob's generation of the shared secret key.
Enter Alice's Public Key (A):
4
Enter Bob's Private Key (b):
3
Enter the Prime number (p):
7
Shared Secret for Bob: 1
Select an Option
1. Encryption
Version Control TODO Problems Terminal Services Build
Build completed successfully in 1 sec, 101 ms (2 minutes ago)
62:18 CRLF UTF-8 4 spaces


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Cryptography Lab File - Diffie_Hellman.java
Cryptography Lab File > Diffie_Hellman > main
Diffie_Hellman.java
Run: Playfair Diffie_Hellman
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Inte
Select an Option
1. Encryption
2. Decryption
3. Brute Force
4. Exit
Enter the choice 1
Welcome to the Diffie-Hellman key exchange algorithm.
Please Enter the Prime number (p):
17
Primitive roots of 17: [3, 5, 6, 7, 10, 11, 12, 14]
Choose a primitive root (g) from the list above:
7
Enter Alice's Private Key (a):
4
Enter Bob's Private Key (b):
3
Public Key of Alice: 4
Public Key of Bob: 3
Shared Secret for Alice: 13
Shared Secret for Bob: 13
Select an Option
1. Encryption
2. Decryption
3. Brute Force
4. Exit
Enter the choice 3
Brute forcing the private keys.
Enter Alice's Public Key (A):
4
Enter Bob's Public Key (B):
3
Enter the Prime number (p):
17
Enter the primitive root (g):
7
Alice's Private Key (a): 4
Bob's Private Key (b): 3
Version Control TODO Problems Terminal Services Build
All files are up-to-date (a minute ago)
47:18 CRLF UTF-8 4 spaces

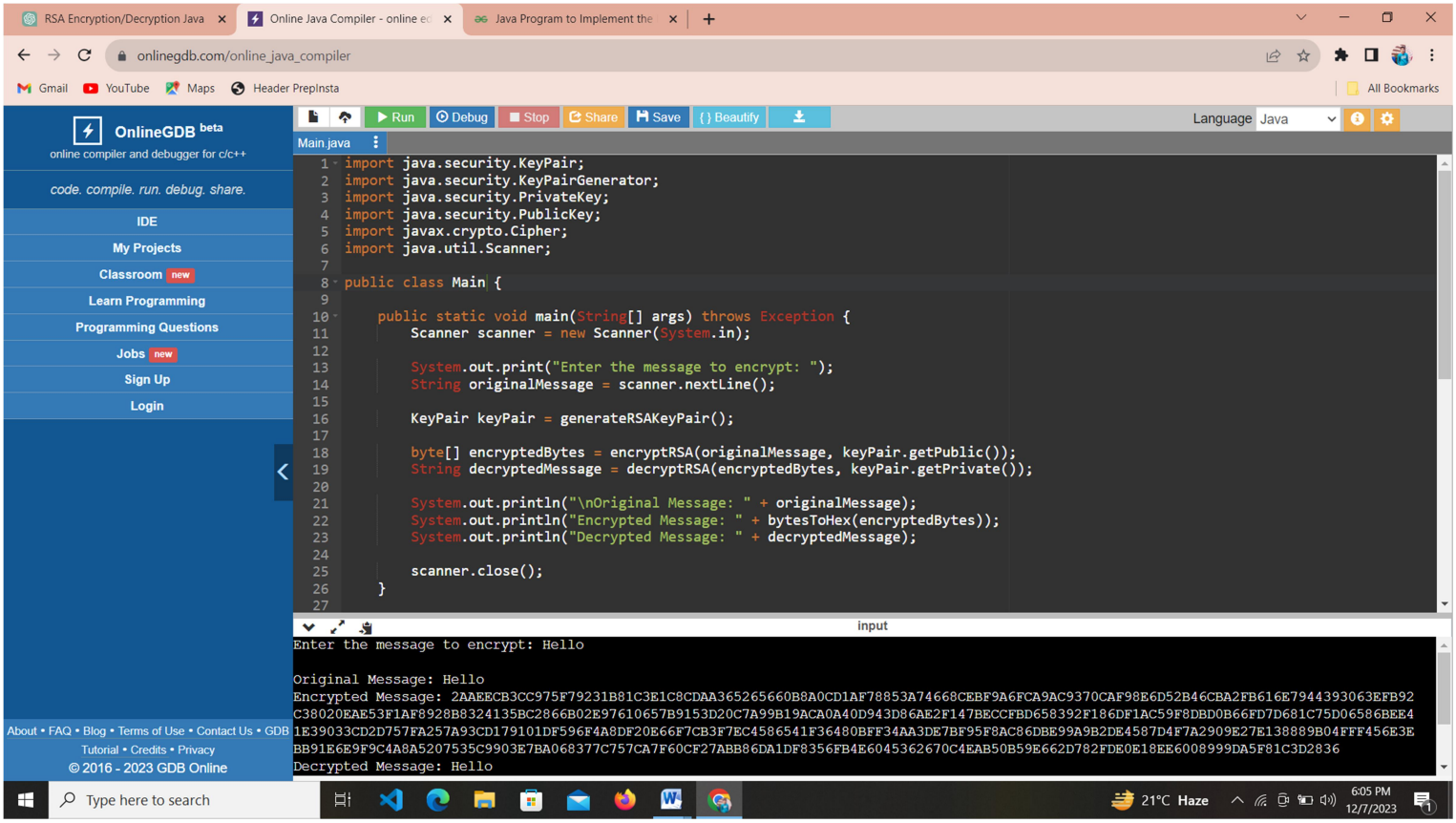
Experiment 7: RSA Algorithm

1. **Write a program to implement RSA Algorithm to generate a pair of keys and show the encryption and decryption by using a given key pair**

Code:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the message to encrypt: ");
        String originalMessage = scanner.nextLine();

        KeyPair keyPair = generateRSAKeyPair();

        byte[] encryptedBytes = encryptRSA(originalMessage, keyPair.getPublic());
        String decryptedMessage = decryptRSA(encryptedBytes, keyPair.getPrivate());

        System.out.println("\nOriginal Message: " + originalMessage);
        System.out.println("Encrypted Message: " + bytesToHex(encryptedBytes));
        System.out.println("Decrypted Message: " + decryptedMessage);

        scanner.close();
    }

    public static KeyPair generateRSAKeyPair() throws Exception {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048); // Key size is 2048 bits
        return keyPairGenerator.generateKeyPair();
    }

    public static byte[] encryptRSA(String plainText, PublicKey publicKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        return cipher.doFinal(plainText.getBytes());
    }

    public static String decryptRSA(byte[] cipherText, PrivateKey privateKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] result = cipher.doFinal(cipherText);
        return new String(result);
    }

    public static String bytesToHex(byte[] bytes) {
        StringBuilder result = new StringBuilder();
        for (byte b : bytes) {
            result.append(String.format("%02X", b));
        }
        return result.toString();
    }
}
```

Output:


RSA Encryption/Decryption Java
Online Java Compiler - online ec
Java Program to Implement the
onlinegdb.com/online_java_compiler
Gmail
YouTube
Maps
Header PrepInsta
All Bookmarks
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login
Run
Debug
Stop
Share
Save
{ } Beautify
Language Java
Main java
1 import java.security.KeyPair;
2 import java.security.KeyPairGenerator;
3 import java.security.PrivateKey;
4 import java.security.PublicKey;
5 import javax.crypto.Cipher;
6 import java.util.Scanner;
7
8 public class Main {
9
10 public static void main(String[] args) throws Exception {
11 Scanner scanner = new Scanner(System.in);
12
13 System.out.print("Enter the message to encrypt: ");
14 String originalMessage = scanner.nextLine();
15
16 KeyPair keyPair = generateRSAKeyPair();
17
18 byte[] encryptedBytes = encryptRSA(originalMessage, keyPair.getPublic());
19 String decryptedMessage = decryptRSA(encryptedBytes, keyPair.getPrivate());
20
21 System.out.println("\nOriginal Message: " + originalMessage);
22 System.out.println("Encrypted Message: " + bytesToHex(encryptedBytes));
23 System.out.println("Decrypted Message: " + decryptedMessage);
24
25 scanner.close();
26 }
27
input
Enter the message to encrypt: Hello
Original Message: Hello
Encrypted Message: 2AAEECB3CC975F79231B81C3E1C8CDAA365265660B8A0CD1AF78853A74668CEBF9A6FCA9AC9370CAF98E6D52B46CBA2FB616E7944393063EFB92C38020EAE53F1AF8928B8324135BC2866B02E97610657B9153D20C7A99B19ACA0A40D943D86AE2F147BECCFBD658392F186DF1AC59F8DBD0B66FD7D681C75D06586BEE41E39033CD2D757FA257A93CD179101DF596F4A8DF20E66F7CB3F7EC4586541F36480BFF34AA3DE7BF95F8AC86DBE99A9B2DE4587D4F7A2909E27E138889B04FFF456E3EBB91E6E9F9C4A8A5207535C9903E7BA068377C757CA7F60CF27ABB86DA1DF8356FB4E6045362670C4EAB50B59E662D782FDE0E18EE6008999DA5F81C3D2836
Decrypted Message: Hello
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2023 GDB Online
Type here to search
21°C Haze
6:05 PM
12/7/2023

Experiment 8: Elgamal Cryptosystem

1.Write a program to implement Elgamal Cryptosystem to generate the pair of keys and then show the encryption & decryption of a given message.

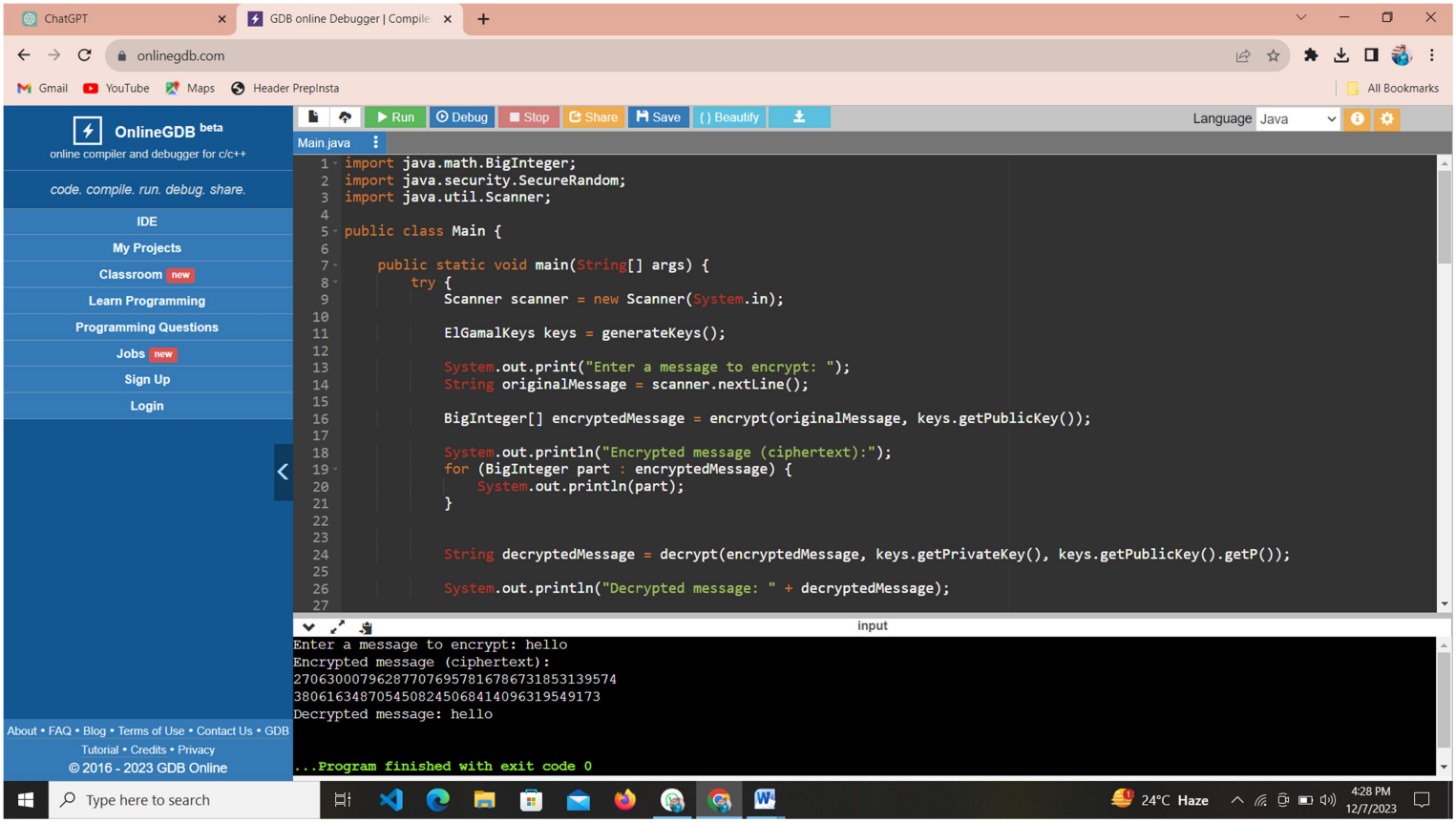
Code:

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);

            ElGamalKeys keys = generateKeys();

            System.out.print("Enter a message to encrypt: ");
            String originalMessage = scanner.nextLine();

            BigInteger[] encryptedMessage = encrypt(originalMessage, keys.getPublicKey());

            System.out.println("Encrypted message (ciphertext):");
            for (BigInteger part : encryptedMessage) {
                System.out.println(part);
            }


            String decryptedMessage = decrypt(encryptedMessage, keys.getPrivateKey(),
keys.getPublicKey().getP());

            System.out.println("Decrypted message: " + decryptedMessage);

            scanner.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static class ElGamalKeys {
        private ElGamalPublicKey publicKey;
        private BigInteger privateKey;

        public ElGamalKeys(ElGamalPublicKey publicKey, BigInteger privateKey) {
            this.publicKey = publicKey;
            this.privateKey = privateKey;
        }

        public ElGamalPublicKey getPublicKey() {
            return publicKey;
        }

        public BigInteger getPrivateKey() {
            return privateKey;
        }
    }
```

```
    static class ElGamalPublicKey {
        private BigInteger p;
        private BigInteger g;
        private BigInteger y;

        public ElGamalPublicKey(BigInteger p, BigInteger g, BigInteger y) {
            this.p = p;
            this.g = g;
            this.y = y;
        }

        public BigInteger getP() {
            return p;
        }

        public BigInteger getG() {
            return g;
        }

        public BigInteger getY() {
            return y;
        }
    }


    public static ElGamalKeys generateKeys() {
        BigInteger p = BigInteger.probablePrime(128, new SecureRandom());
        BigInteger g = BigInteger.valueOf(2);


        BigInteger privateKey = new BigInteger(p.bitCount() - 2, new SecureRandom());

        BigInteger publicKey = g.modPow(privateKey, p);

        return new ElGamalKeys(new ElGamalPublicKey(p, g, publicKey), privateKey);
    }

    public static BigInteger[] encrypt(String message, ElGamalPublicKey publicKey) {
        BigInteger p = publicKey.getP();
        BigInteger g = publicKey.getG();
        BigInteger y = publicKey.getY();

        SecureRandom random = new SecureRandom();
        BigInteger k = new BigInteger(p.bitCount() - 2, random);

        BigInteger m = new BigInteger(message.getBytes());


        BigInteger c1 = g.modPow(k, p);
        BigInteger c2 = y.modPow(k, p).multiply(m).mod(p);

        return new BigInteger[]{c1, c2};
    }


    public static String decrypt(BigInteger[] encryptedMessage, BigInteger privateKey, BigInteger
p) {
        BigInteger c1 = encryptedMessage[0];
        BigInteger c2 = encryptedMessage[1];

        BigInteger s = c1.modPow(privateKey, p);

        BigInteger sInverse = s.modInverse(p);
```

```
        BigInteger decrypted = c2.multiply(sInverse).mod(p);

        return new String(decrypted.toByteArray());
    }
}
```

Output:



Experiment 9: Rabbin-Miller Test

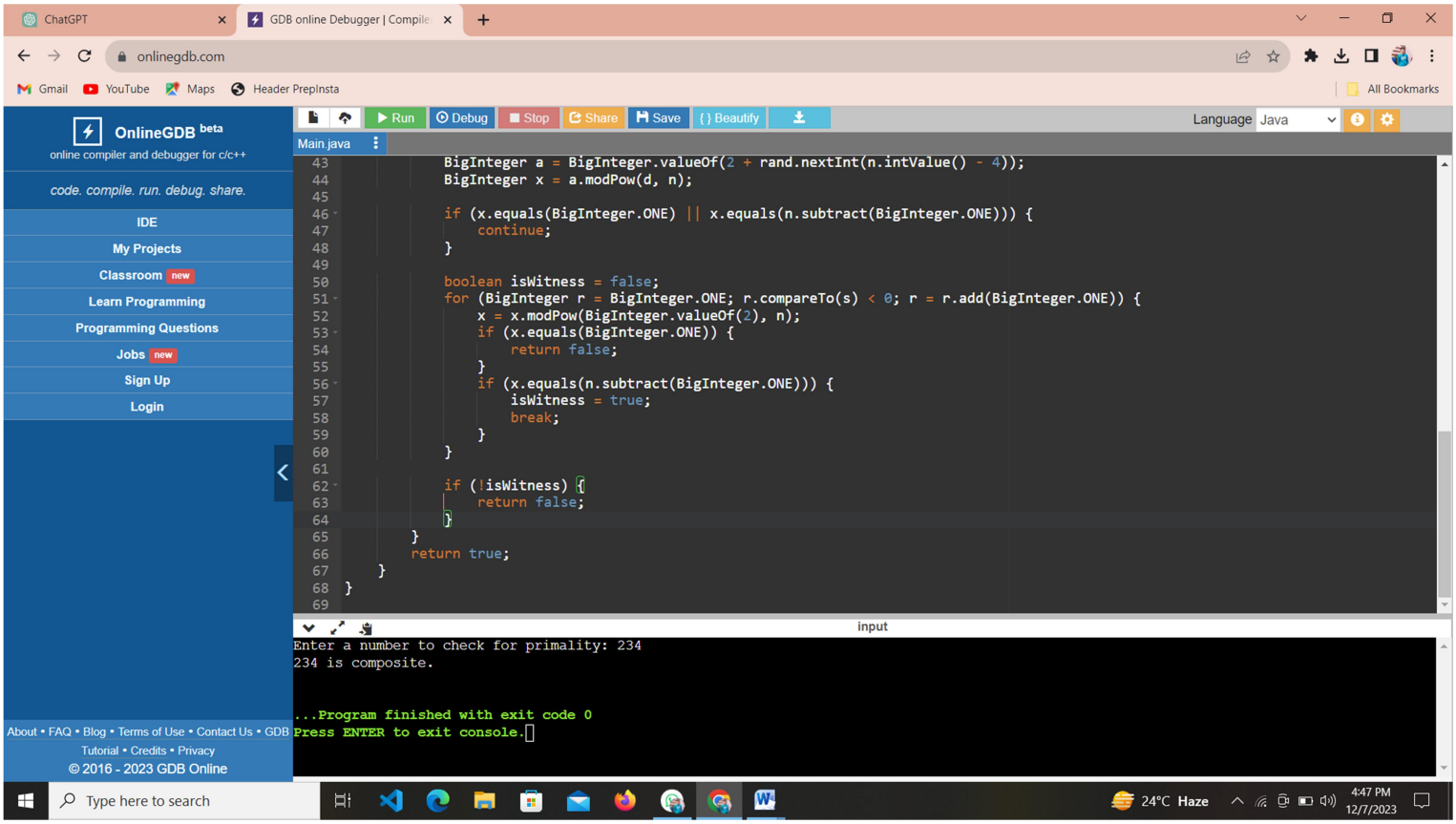
1. Write a program to implement Rabin Miller Primality Test to checkwhether given number is prime or composite

Code:

```
import java.util.Scanner;
import java.math.BigInteger;
import java.security.SecureRandom;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number to check for primality: ");
        BigInteger number = scanner.nextBigInteger();

        if (isPrime(number, 50)) {
            System.out.println(number + " is likely a prime number.");
        } else {
            System.out.println(number + " is composite.");
        }

        scanner.close();
    }

    public static boolean isPrime(BigInteger n, int k) {
        if (n.compareTo(BigInteger.valueOf(2)) < 0) {
            return false;
        }
        if (n.compareTo(BigInteger.valueOf(2)) == 0 || n.compareTo(BigInteger.valueOf(3)) == 0) {
            return true;
        }
        if (n.mod(BigInteger.valueOf(2)).equals(BigInteger.ZERO)) {
            return false;
        }

        BigInteger s = BigInteger.ZERO;
        BigInteger d = n.subtract(BigInteger.ONE);

        while (d.mod(BigInteger.valueOf(2)).equals(BigInteger.ZERO)) {
            s = s.add(BigInteger.ONE);
            d = d.divide(BigInteger.valueOf(2));
        }

        SecureRandom rand = new SecureRandom();
        for (int i = 0; i < k; i++) {
            BigInteger a = BigInteger.valueOf(2 + rand.nextInt(n.intValue() - 4));
            BigInteger x = a.modPow(d, n);

            if (x.equals(BigInteger.ONE) || x.equals(n.subtract(BigInteger.ONE))) {
                continue;
            }

            boolean isWitness = false;
            for (BigInteger r = BigInteger.ONE; r.compareTo(s) < 0; r = r.add(BigInteger.ONE)) {
                x = x.modPow(BigInteger.valueOf(2), n);
                if (x.equals(BigInteger.ONE)) {
                    return false;
                }
                if (x.equals(n.subtract(BigInteger.ONE))) {
```

```
                    isWitness = true;
                    break;
                }
            }

            if (!isWitness) {
                return false;
            }
        }
        return true;
    }
}
```

Output:



Experiment 10: Hill Cipher

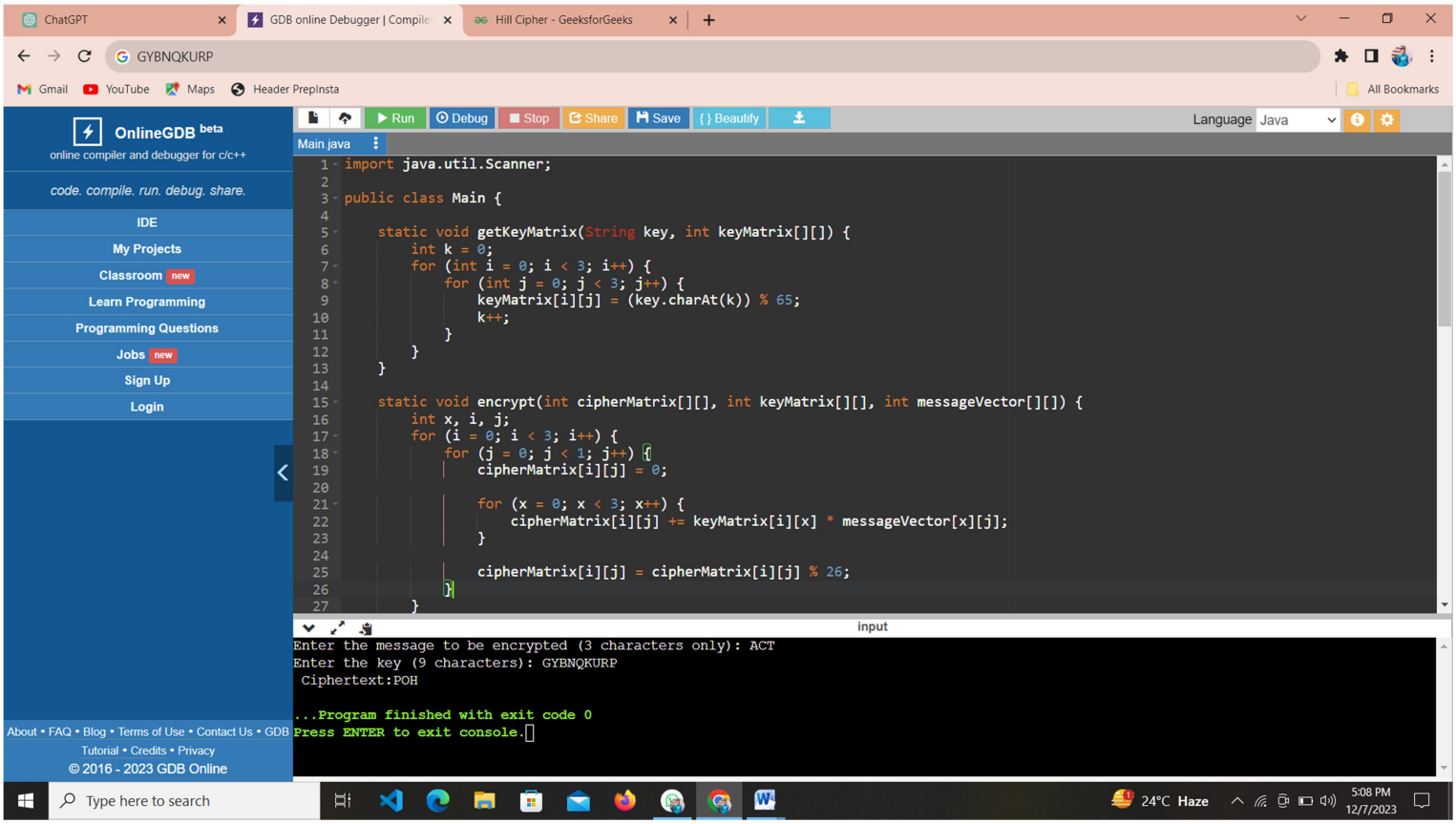
1 **Write a program to implement Hill Cipher to encrypt & decrypt the given message by using a given key matrix. Show the values for key and its corresponding key inverse values.**

Code:

```
import java.util.Scanner;

public class Main {

    static void getKeyMatrix(String key, int keyMatrix[][]) {
        int k = 0;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                keyMatrix[i][j] = (key.charAt(k)) % 65;
                k++;
            }
        }
    }

    static void encrypt(int cipherMatrix[][], int keyMatrix[][], int messageVector[][]) {
        int x, i, j;
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 1; j++) {
                cipherMatrix[i][j] = 0;

                for (x = 0; x < 3; x++) {
                    cipherMatrix[i][j] += keyMatrix[i][x] * messageVector[x][j];
                }

                cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
            }
        }
    }

    static void HillCipher(String message, String key) {
        int[][] keyMatrix = new int[3][3];
        getKeyMatrix(key, keyMatrix);

        int[][] messageVector = new int[3][1];

        for (int i = 0; i < 3; i++)
            messageVector[i][0] = (message.charAt(i)) % 65;

        int[][] cipherMatrix = new int[3][1];
        encrypt(cipherMatrix, keyMatrix, messageVector);

        String CipherText = "";

        for (int i = 0; i < 3; i++)
            CipherText += (char) (cipherMatrix[i][0] + 65);

        System.out.print(" Ciphertext:" + CipherText);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the message to be encrypted (3 characters only): ");
        String message = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "");

        while (message.length() != 3) {
```

```
            System.out.println("Please enter exactly 3 characters.");
            message = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "");
        }

        System.out.print("Enter the key (9 characters): ");
        String key = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "");

        while (key.length() != 9) {
            System.out.println("Please enter exactly 9 characters.");
            key = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "");
        }

        HillCipher(message, key);

        scanner.close();
    }
}
```

Output:



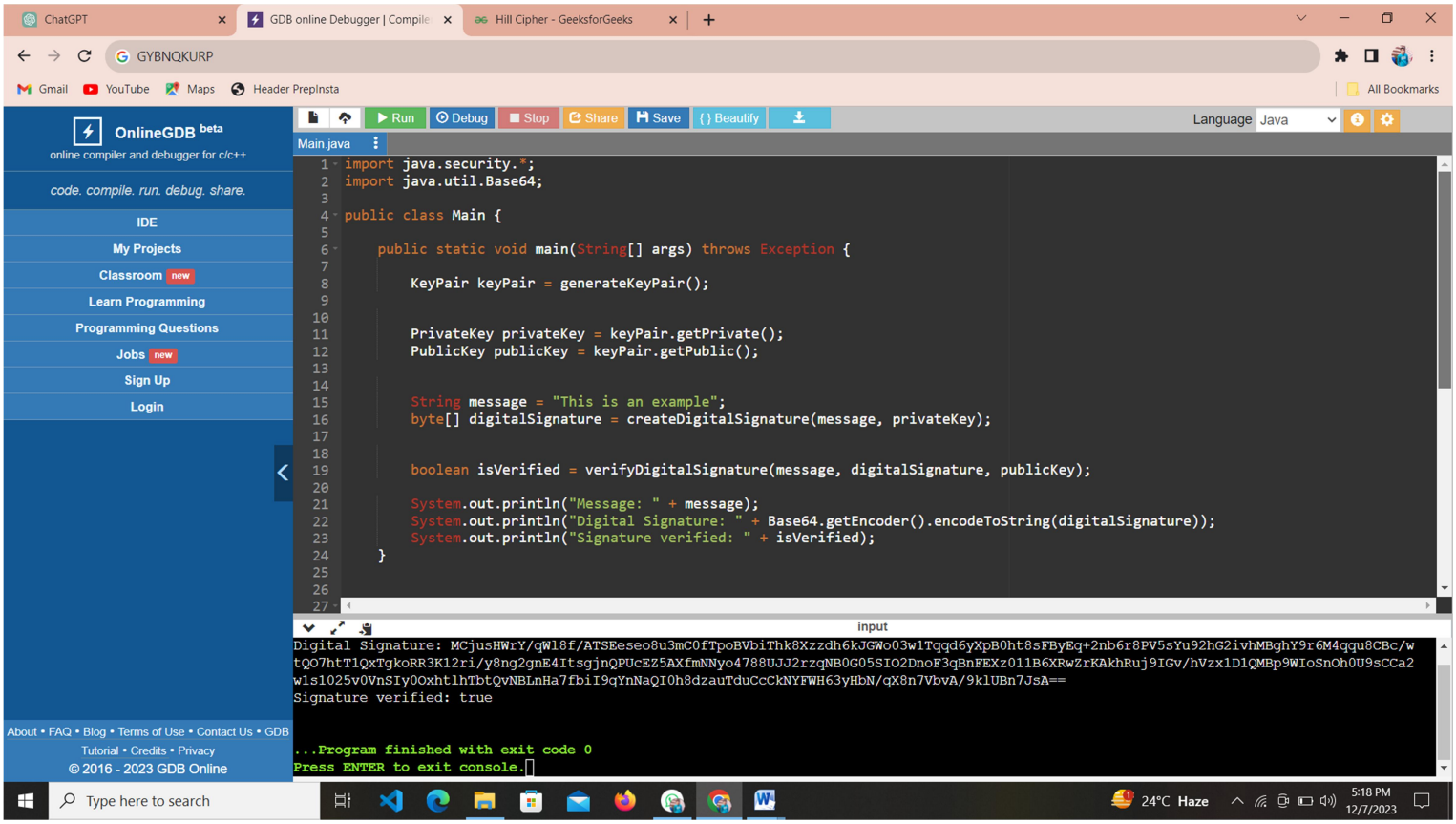
Experiment 11: Digital Signature

1.Write a Program to implement Alice is of one of the employee of company XYZ. He wants to ensure that whatever data he is sending to Bob should be checked for accuracy. Implement the RSA digital signature for the message "This is an example" for showing the Digital Signature in such a scenario.

Code:

```
import java.security.*;
import java.util.Base64;

public class Main {

    public static void main(String[] args) throws Exception {

        KeyPair keyPair = generateKeyPair();


        PrivateKey privateKey = keyPair.getPrivate();
        PublicKey publicKey = keyPair.getPublic();


        String message = "This is an example";
        byte[] digitalSignature = createDigitalSignature(message, privateKey);


        boolean isVerified = verifyDigitalSignature(message, digitalSignature, publicKey);

        System.out.println("Message: " + message);
        System.out.println("Digital Signature: " +
Base64.getEncoder().encodeToString(digitalSignature));
        System.out.println("Signature verified: " + isVerified);
    }


    public static KeyPair generateKeyPair() throws Exception {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048);
        return keyPairGenerator.generateKeyPair();
    }


    public static byte[] createDigitalSignature(String message, PrivateKey privateKey) throws
Exception {
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initSign(privateKey);
        signature.update(message.getBytes());
        return signature.sign();
    }


    public static boolean verifyDigitalSignature(String message, byte[] signatureToVerify,
PublicKey publicKey) throws Exception {
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initVerify(publicKey);
        signature.update(message.getBytes());
        return signature.verify(signatureToVerify);
    }
}
```

Output:


ChatGPT
GDB online Debugger | Compile
Hill Cipher - GeeksforGeeks
GYBNQKURP
Gmail
YouTube
Maps
Header PrepInsta
All Bookmarks
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2023 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language Java
Main.java
1 import java.security.*;
2 import java.util.Base64;
3
4 public class Main {
5
6 public static void main(String[] args) throws Exception {
7
8 KeyPair keyPair = generateKeyPair();
9
10
11 PrivateKey privateKey = keyPair.getPrivate();
12 PublicKey publicKey = keyPair.getPublic();
13
14
15 String message = "This is an example";
16 byte[] digitalSignature = createDigitalSignature(message, privateKey);
17
18
19 boolean isVerified = verifyDigitalSignature(message, digitalSignature, publicKey);
20
21 System.out.println("Message: " + message);
22 System.out.println("Digital Signature: " + Base64.getEncoder().encodeToString(digitalSignature));
23 System.out.println("Signature verified: " + isVerified);
24 }
25
26
27
input
Digital Signature: MCjusHWrY/qWl8f/ATSEeseo8u3mC0fTpoBVbiThk8Xzzdh6kJGWo03w1Tqqd6yXpB0ht8sFByEq+2nb6r8PV5sYu92hG2ivhMBghY9r6M4qqu8CBc/w
tQO7htT1QxTgkoRR3K12ri/y8ng2gnE4ItsgjnQPUcEZ5AXfmNNyo4788UJJ2rzqNB0G05SIO2DnoF3qBnFEXz011B6XRwZrKAkhRuj9IGv/hVzx1D1QMBp9WIoSnOh0U9sCCa2
w1s1025v0VnSIy0OxhtlhTbtQvNBLnHa7fbiI9qYnNaQI0h8dzauTduCcCkNYFWH63yHbN/qX8n7VbvA/9klUBn7JsA==
Signature verified: true
...Program finished with exit code 0
Press ENTER to exit console.
Type here to search
24°C Haze
5:18 PM
12/7/2023

Experiment 12: Digital Signature

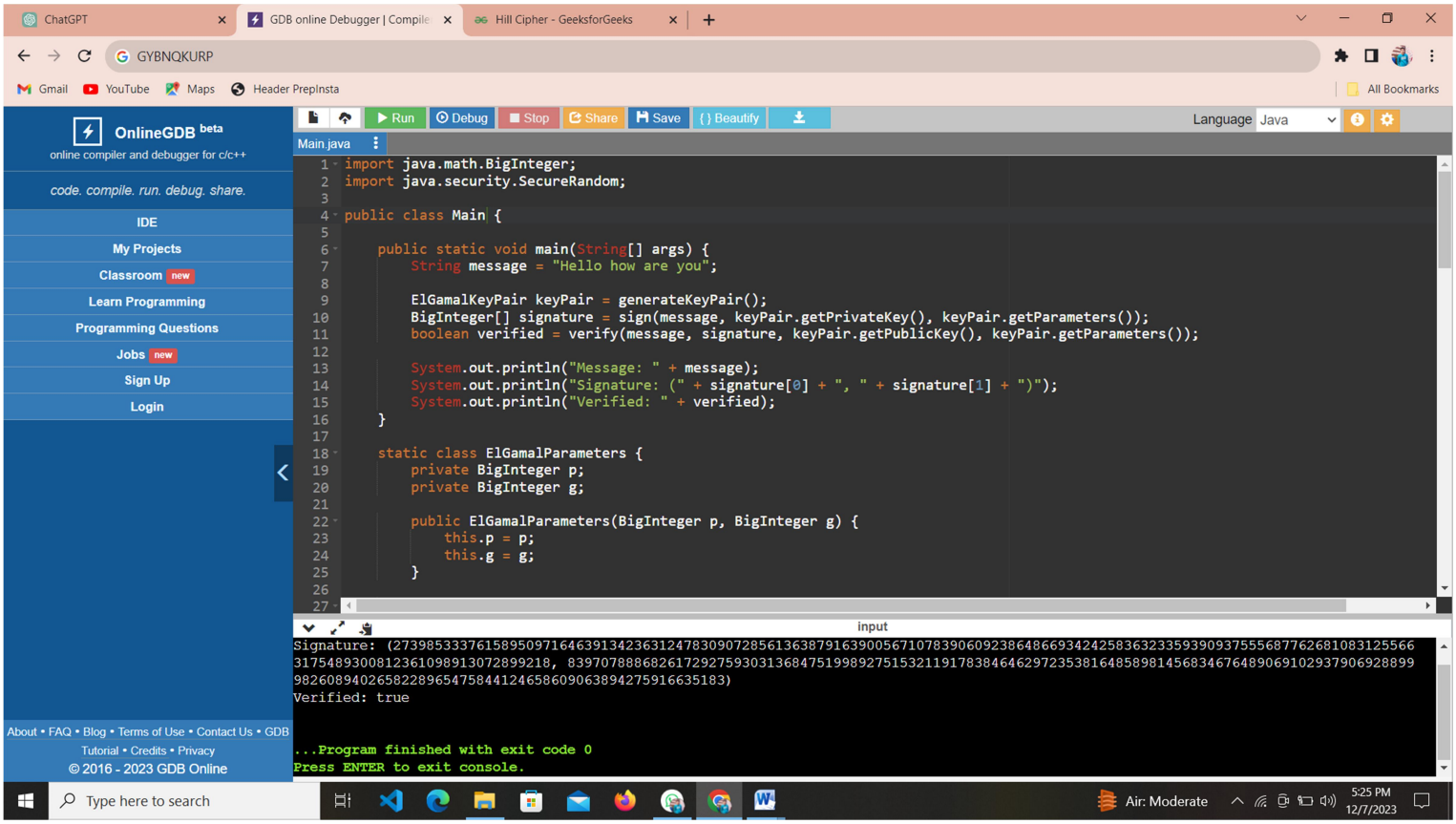
1. **Alice is of one of the employee of company XYZ. He wants to ensure that whatever data he is sending to Bob should be checked for accuracy. Implement the Elgamal digital signature for the message "Hello how are you" for showing the Digital Signature in such a scenario.**

Code:

```
import java.math.BigInteger;
import java.security.SecureRandom;

public class Main {

    public static void main(String[] args) {
        String message = "Hello how are you";

        ElGamalKeyPair keyPair = generateKeyPair();
        BigInteger[] signature = sign(message, keyPair.getPrivateKey(), keyPair.getParameters());
        boolean verified = verify(message, signature, keyPair.getPublicKey(),
keyPair.getParameters());

        System.out.println("Message: " + message);
        System.out.println("Signature: (" + signature[0] + ", " + signature[1] + ")");
        System.out.println("Verified: " + verified);
    }

    static class ElGamalParameters {
        private BigInteger p;
        private BigInteger g;

        public ElGamalParameters(BigInteger p, BigInteger g) {
            this.p = p;
            this.g = g;
        }

        public BigInteger getP() {
            return p;
        }

        public BigInteger getG() {
            return g;
        }
    }

    static class ElGamalKeyPair {
        private BigInteger privateKey;
        private BigInteger publicKey;
        private ElGamalParameters parameters;

        public ElGamalKeyPair(BigInteger privateKey, BigInteger publicKey, ElGamalParameters
parameters) {
            this.privateKey = privateKey;
            this.publicKey = publicKey;
            this.parameters = parameters;
        }

        public BigInteger getPrivateKey() {
            return privateKey;
        }
```

```
        public BigInteger getPublicKey() {
            return publicKey;
        }

        public ElGamalParameters getParameters() {
            return parameters;
        }
    }

    public static ElGamalKeyPair generateKeyPair() {
        SecureRandom random = new SecureRandom();

        BigInteger p = new BigInteger(512, 64, random);
        BigInteger g = new BigInteger(512, random).mod(p.subtract(BigInteger.ONE));

        BigInteger privateKey = new BigInteger(512, random).mod(p.subtract(BigInteger.ONE));
        BigInteger publicKey = g.modPow(privateKey, p);

        ElGamalParameters parameters = new ElGamalParameters(p, g);
        return new ElGamalKeyPair(privateKey, publicKey, parameters);
    }

    public static BigInteger[] sign(String message, BigInteger privateKey, ElGamalParameters
parameters) {
        BigInteger p = parameters.getP();
        BigInteger g = parameters.getG();

        SecureRandom random = new SecureRandom();
        BigInteger k;
        do {
            k = new BigInteger(p.bitLength(), random);
        } while (k.compareTo(BigInteger.ZERO) <= 0 || k.compareTo(p.subtract(BigInteger.ONE)) >=
0);

        BigInteger h = new BigInteger(message.getBytes()).mod(p.subtract(BigInteger.ONE));

        BigInteger r = g.modPow(k, p);
        BigInteger s =
k.modInverse(p.subtract(BigInteger.ONE)).multiply(h.subtract(privateKey.multiply(r))).mod(p.subtrac
t(BigInteger.ONE));

        return new BigInteger[]{r, s};
    }

    public static boolean verify(String message, BigInteger[] signature, BigInteger publicKey,
ElGamalParameters parameters) {
        BigInteger p = parameters.getP();
        BigInteger g = parameters.getG();
        BigInteger r = signature[0];
        BigInteger s = signature[1];

        if (r.compareTo(BigInteger.ONE) < 0 || r.compareTo(p.subtract(BigInteger.ONE)) > 0 ||
                s.compareTo(BigInteger.ONE) < 0 || s.compareTo(p.subtract(BigInteger.ONE)) > 0) {
            return false;
        }

        BigInteger h = new BigInteger(message.getBytes()).mod(p.subtract(BigInteger.ONE));

        BigInteger v1 = g.modPow(h, p);
        BigInteger v2 = publicKey.modPow(r, p).multiply(r.modPow(s, p)).mod(p);

        return v1.equals(v2);
    }
}
```

Output:

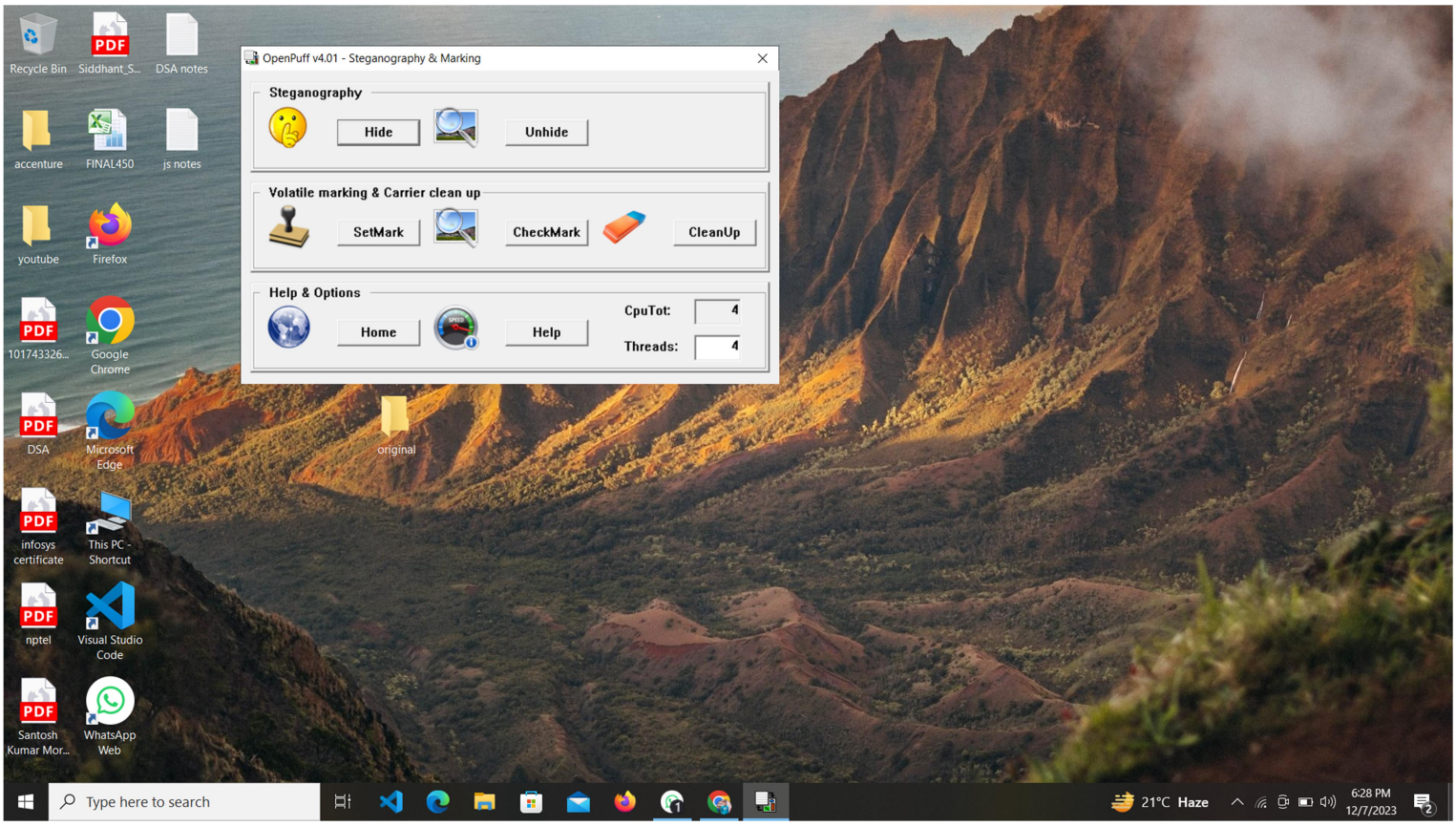

ChatGPT
GDB online Debugger | Compile
Hill Cipher - GeeksforGeeks
GYBNQKURP
Gmail
YouTube
Maps
Header PrepInsta
All Bookmarks
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2023 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language Java
Main java
1 import java.math.BigInteger;
2 import java.security.SecureRandom;
3
4 public class Main {
5
6 public static void main(String[] args) {
7 String message = "Hello how are you";
8
9 ElGamalKeyPair keyPair = generateKeyPair();
10 BigInteger[] signature = sign(message, keyPair.getPrivateKey(), keyPair.getParameters());
11 boolean verified = verify(message, signature, keyPair.getPublicKey(), keyPair.getParameters());
12
13 System.out.println("Message: " + message);
14 System.out.println("Signature: (" + signature[0] + ", " + signature[1] + ")");
15 System.out.println("Verified: " + verified);
16 }
17
18 static class ElGamalParameters {
19 private BigInteger p;
20 private BigInteger g;
21
22 public ElGamalParameters(BigInteger p, BigInteger g) {
23 this.p = p;
24 this.g = g;
25 }
26
27
input
Signature: (27398533376158950971646391342363124783090728561363879163900567107839060923864866934242583632335939093755568776268108312556631754893008123610989130728992218, 83970788868261729275930313684751998927515321191783846462972353816485898145683467648906910293790692889989826089402658228965475844124658609063894275916635183)
Verified: true
...Program finished with exit code 0
Press ENTER to exit console.
Type here to search
Air: Moderate
5:25 PM
12/7/2023

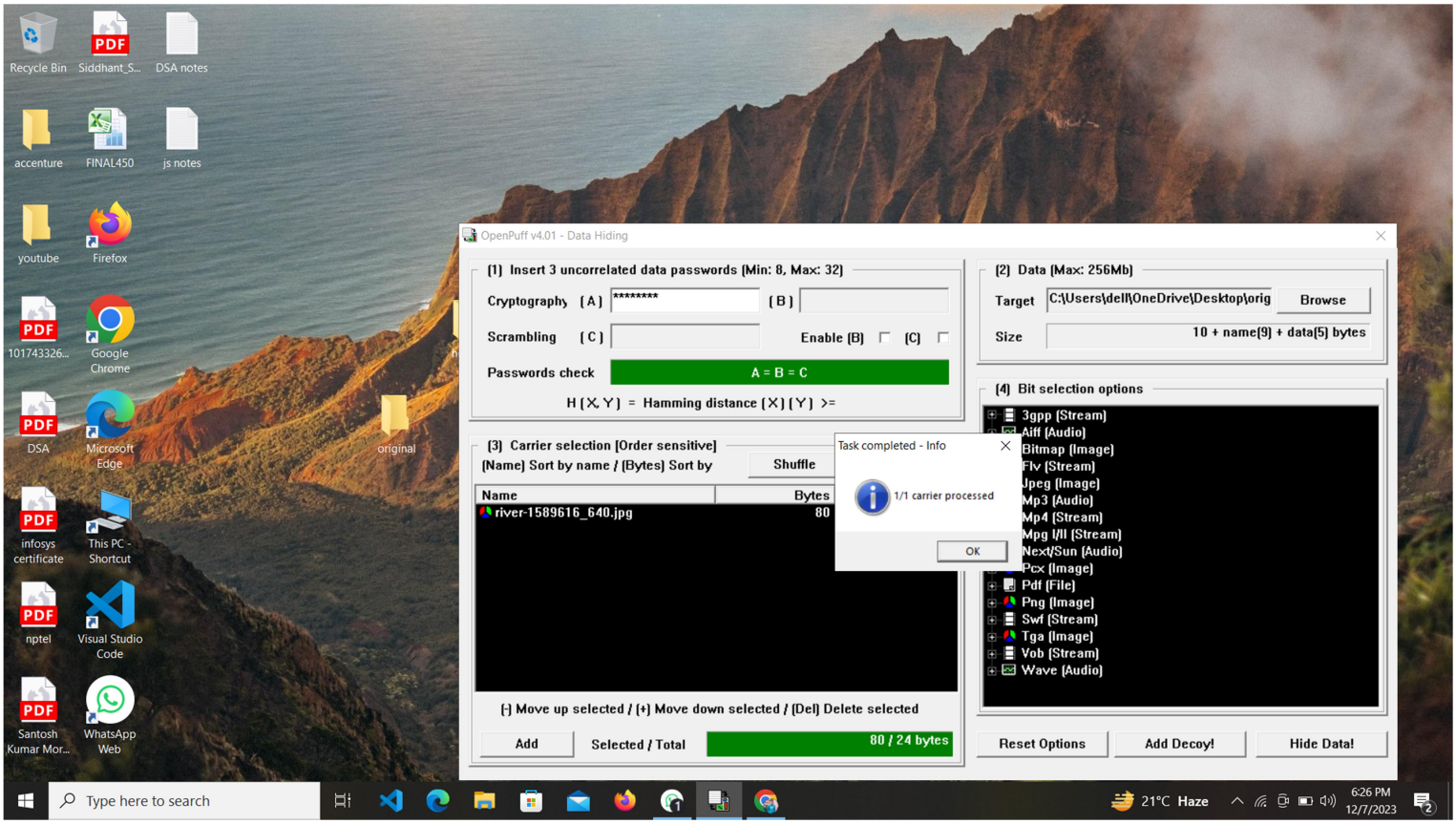
Experiment 13: Digital Signature

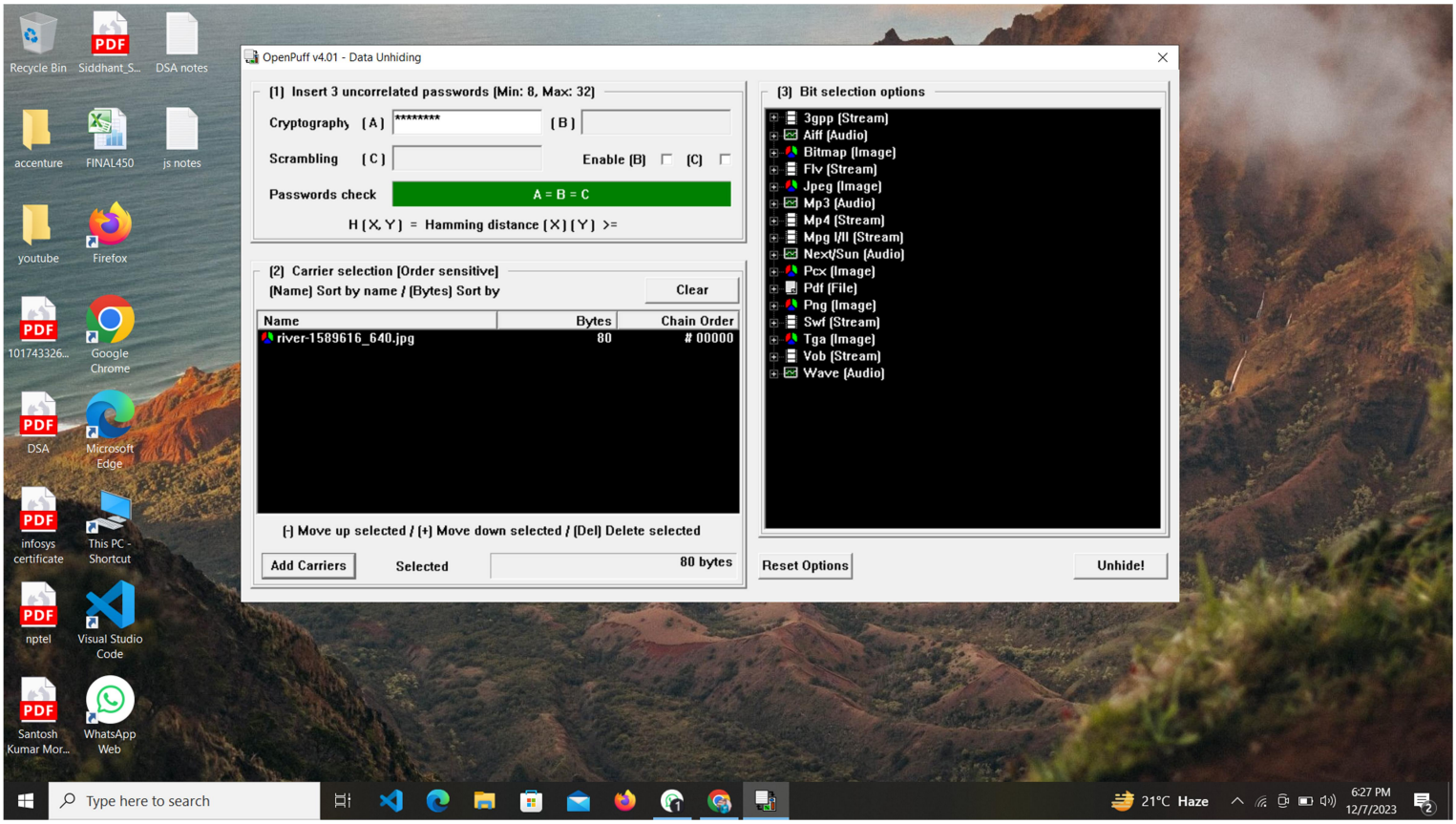
1. **Steganography using openpuff tool.**

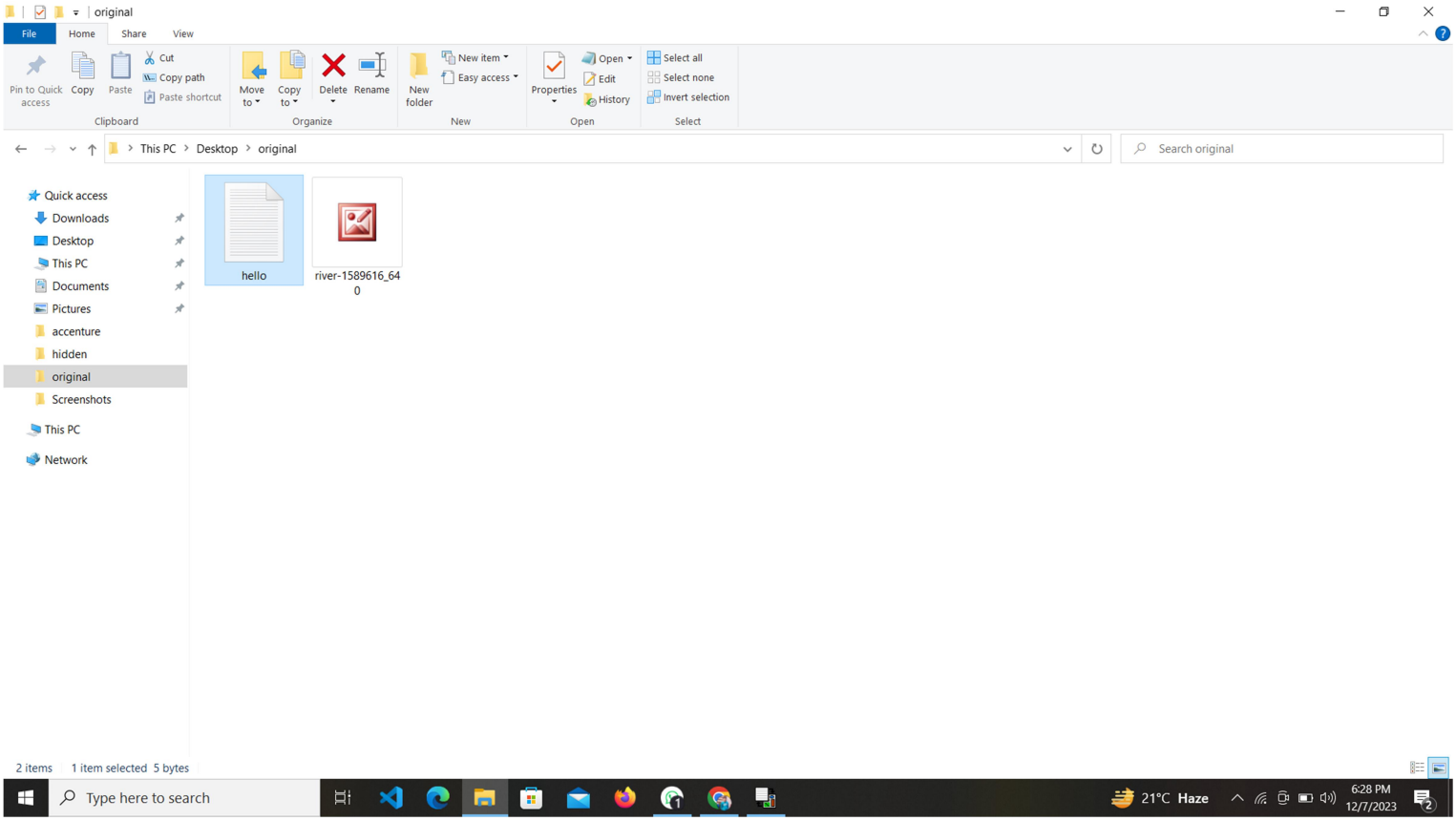
Objective: Students will explore how to hide data using the concept of Steganography in multiple carrier file.

Preview:




OpenPuff v4.01 - Data Hiding
(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)
Cryptography (A) ********
(B)
Scrambling (C)
Enable (B) (C)
Passwords check
A = B = C
H (X, Y) = Hamming distance (X) (Y) >=
(2) Data (Max: 256Mb)
Target C:\Users\dell\OneDrive\Desktop\orig
Browse
Size
10 + name(9) + data(5) bytes
(3) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by
Shuffle
Name
Bytes
river-1589616_640.jpg
80
(-) Move up selected / (+) Move down selected / (Del) Delete selected
Add
Selected / Total
80 / 24 bytes
Task completed - Info
1/1 carrier processed
OK
(4) Bit selection options
3gpp (Stream)
Aiff (Audio)
Bitmap (Image)
Flv (Stream)
Jpeg (Image)
Mp3 (Audio)
Mp4 (Stream)
Mpg I/II (Stream)
Next/Sun (Audio)
Pcx (Image)
Pdf (File)
Png (Image)
Swf (Stream)
Tga (Image)
Vob (Stream)
Wave (Audio)
Reset Options
Add Decoy!
Hide Data!
Type here to search
21°C Haze
6:26 PM
12/7/2023


OpenPuff v4.01 - Data Unhiding
(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)
Cryptography (A) ********
(B)
Scrambling (C)
Enable (B) (C)
Passwords check
A = B = C
H (X, Y) = Hamming distance (X) (Y) >=
(2) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by
Clear
Name
Bytes
Chain Order
river-1589616_640.jpg
80
00000
(-) Move up selected / (+) Move down selected / (Del) Delete selected
Add Carriers
Selected
80 bytes
(3) Bit selection options
3gpp (Stream)
Aiff (Audio)
Bitmap (Image)
Flv (Stream)
Jpeg (Image)
Mp3 (Audio)
Mp4 (Stream)
Mpg I/II (Stream)
Next/Sun (Audio)
Pcx (Image)
Pdf (File)
Png (Image)
Swf (Stream)
Tga (Image)
Vob (Stream)
Wave (Audio)
Reset Options
Unhide!
Type here to search
21°C Haze
6:27 PM
12/7/2023


original
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC
Desktop
original
Search original
Quick access
Downloads
Desktop
This PC
Documents
Pictures
accenture
hidden
original
Screenshots
This PC
Network
hello
river-1589616_640
2 items
1 item selected 5 bytes
Type here to search
21°C Haze
6:28 PM
12/7/2023