Term Work 1

```
package termwork_1;

import java.io.*;
import java.nio.file.*;
import java.security.*;
import java.util.Scanner;

public class TermWork_1 {

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

        System.out.println("Enter the file path:");
        String filePath = sc.nextLine();

        System.out.println("Enter the path to store/read hash:");
        String hashFilePath = sc.nextLine();

        try {
            String currentHash = computeHash(filePath);
            System.out.println("Current Hash: " + currentHash);

            Path hashPath = Paths.get(hashFilePath);

            if (Files.exists(hashPath)) {
                String storedHash = Files.readString(hashPath);
                System.out.println("Stored Hash: " + storedHash);

                if (currentHash.equals(storedHash))
                    System.out.println("File integrity verified.");
                else
                    System.out.println("File integrity check failed! File altered.");
            } else {
                Files.writeString(hashPath, currentHash);
                System.out.println("Hash stored successfully.");
            }

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }

        sc.close();
    }
    private static String computeHash(String filePath) throws Exception {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        try (InputStream is = new FileInputStream(filePath)) {
            byte[] buffer = new byte[1024];
            int bytesRead;

            while ((bytesRead = is.read(buffer)) != -1)
                digest.update(buffer, 0, bytesRead);
        }

        StringBuilder hash = new StringBuilder();
        for (byte b : digest.digest())
            hash.append(String.format("%02x", b));

        return hash.toString();
    }
}
```

Term Work 2

```
package termwork_2;

import java.util.*;

public class TermWork_2 {

    public static void main(String[] args) {

        List<WiFiNetwork> networks = scanWiFiNetworks();

        if (networks.isEmpty()) {
            System.out.println("No Wi-Fi networks found nearby.");
            return;
        }

        System.out.println("Wi-Fi networks found:");
        networks.forEach(System.out::println);

        // Sort by signal strength (descending)
        networks.sort(Comparator.comparingInt(WiFiNetwork::signalStrength).reversed());

        System.out.println("\nWi-Fi networks sorted by signal strength:");
        networks.forEach(System.out::println);
    }

    static List<WiFiNetwork> scanWiFiNetworks() {
        return new ArrayList<>(List.of(
            new WiFiNetwork("WiFi Network 1", "00:11:22:33:44:55", -70),
            new WiFiNetwork("WiFi Network 2", "11:22:33:44:55:66", -80),
            new WiFiNetwork("WiFi Network 3", "22:33:44:55:66:77", -60)
        ));
    }

    record WiFiNetwork(String ssid, String bssid, int signalStrength) {
        @Override
        public String toString() {
            return "SSID: " + ssid + ", BSSID: " + bssid + ", Signal: " + signalStrength + " dBm";
        }
    }
}
```

Term Work 3

```
package termwork_3_cs;
import java.util.*;

public class TermWork_3_CS {

    private final Set<String> allowedIPs = new HashSet<>();

    public boolean isAllowed(String ip) {
        return allowedIPs.contains(ip);
    }

    public static void main(String[] args) {
        TermWork_3_CS firewall = new TermWork_3_CS();
        Scanner sc = new Scanner(System.in);

        // Pre-added allowed IPs
        firewall.allowedIPs.addAll(List.of(
            "192.168.1.1",
            "192.168.1.2",
            "10.0.0.1"
        ));

        while (true) {
            System.out.print("Enter IP to check (or 'exit'): ");
            String ip = sc.nextLine();

            if (ip.equalsIgnoreCase("exit")) break;

            System.out.println(firewall.isAllowed(ip)
                ? "IP " + ip + " is allowed."
                : "IP " + ip + " is blocked.");
        }

        sc.close();
    }
}
```

Term Work 4

```
package termwork_4_cs;

import java.security.SecureRandom;
import java.util.*;

public class TermWork_4_CS {

   private static final String U = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   private static final String L = "abcdefghijklmnopqrstuvwxyz";
   private static final String D = "0123456789";
   private static final String S = "!@#$%^&*()-_=+<>?";
   private static final String ALL = U + L + D + S;

   private static final SecureRandom R = new SecureRandom();

   public static void main(String[] args) {
      System.out.println("Generated Password: " + generate(12));
   }

   public static String generate(int len) {
      if (len < 4) throw new IllegalArgumentException("Min length: 4");

      List<Character> pass = new ArrayList<>(List.of(
         rand(U), rand(L), rand(D), rand(S)
      ));

      for (int i = 4; i < len; i++)
         pass.add(rand(ALL));

      Collections.shuffle(pass, R);

      StringBuilder sb = new StringBuilder(len);
      pass.forEach(sb::append);
      return sb.toString();
   }

   private static char rand(String src) {
      return src.charAt(R.nextInt(src.length()));
   }
}
```

Term Work 5

```
package termwork_5_cs;

import java.net.*;

public class TermWork_5_CS {

    public static void main(String[] args) {
        String ip = "127.0.0.1";
        System.out.println("Scanning ports on " + ip);

        for (int port = 1; port <= 1024; port++) {
            try (Socket s = new Socket(ip, port)) {
                System.out.println("Port " + port + " is open.");
            } catch (Exception ignored) {}
        }

        System.out.println("Scan complete.");
    }
}
```

Term Work 6

```
package termwork_6_cs;

import java.util.Scanner;

public class TermWork_6_CS {

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

        System.out.print("Enter text: ");
        String text = sc.nextLine();

        System.out.print("Enter shift: ");
        int shift = sc.nextInt();

        String enc = cipher(text, shift);
        System.out.println("Encrypted: " + enc);
        System.out.println("Decrypted: " + cipher(enc, 26 - shift));

        sc.close();
    }

    static String cipher(String text, int shift) {
        StringBuilder sb = new StringBuilder();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                char base = Character.isUpperCase(c) ? 'A' : 'a';
                sb.append((char) ((c - base + shift) % 26 + base));
            } else sb.append(c);
        }
        return sb.toString();
    }
}
```

Term Work 7

```
package termwork_7_cs;

import java.security.*;

public class TermWork_7_CS {

    public static void main(String[] args) {
        try {
            KeyPairGenerator gen = KeyPairGenerator.getInstance("RSA");
            gen.initialize(2048);
            KeyPair pair = gen.generateKeyPair();

            String msg = "Hello, this is a message to be signed.";
            byte[] data = msg.getBytes();

            // Sign
            Signature sign = Signature.getInstance("SHA256withRSA");
            sign.initSign(pair.getPrivate());
            sign.update(data);
            byte[] signature = sign.sign();
            System.out.println("Digital Signature generated.");

            // Verify
            Signature verify = Signature.getInstance("SHA256withRSA");
            verify.initVerify(pair.getPublic());
            verify.update(data);

            System.out.println(verify.verify(signature) ?
                    "Signature verified. Message is authentic." :
                    "Signature verification failed.");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Term Work 8

```
package termwork_8_cs;

import java.util.*;

public class TermWork_8_CS {

    private static final Set<String> phishing = new HashSet<>(Arrays.asList(
            "http://example-phishing1.com",
            "http://example-phishing2.com",
            "http://example-phishing3.com"
    ));

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

        System.out.print("Enter URL: ");
        String url = sc.nextLine();

        System.out.println(phishing.contains(url)
                ? "Alert: The site is malicious!"
                : "The site is safe.");

        sc.close();
    }
}
```