Bài 1

|  |
| --- |
| +---------------------------+  | Nhập email và số điện thoại |  +---------------------------+  |  +--------------------------+  | Kiểm tra định dạng email |  +--------------------------+  |  +---------------------------------+  | Email hợp lệ? (Đúng định dạng) |  +---------+-----------+----------+  | |  +------+ +------+  | Có | Không  | |  +--------v--------+ +--------v--------+  | Kiểm tra số ĐT | | Thông báo lỗi |  +-----------------+ +-----------------+  |  +------------------------+  | SĐT hợp lệ? (đúng độ dài) |  +--------+--------+  |  +-----v-------+  | Thông báo OK |  +-------------+ |

Code

|  |
| --- |
| import java.util.Scanner;  import java.util.regex.Pattern;  public class InputValidation {  // Kiểm tra định dạng email với regex  public static boolean isValidEmail(String email) {  String emailRegex = "^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";  Pattern pattern = Pattern.compile(emailRegex);  return pattern.matcher(email).matches();  }  // Kiểm tra số điện thoại chỉ chứa số và có độ dài 10 chữ số  public static boolean isValidPhoneNumber(String phoneNumber) {  return phoneNumber.matches("\\d{10}"); // Kiểm tra số điện thoại có đúng 10 số  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Nhập email từ người dùng  System.out.print("Nhập địa chỉ email: ");  String email = scanner.nextLine();  // Kiểm tra email hợp lệ  if (!isValidEmail(email)) {  System.out.println("Email không hợp lệ.");  return;  }  // Nhập số điện thoại từ người dùng  System.out.print("Nhập số điện thoại (10 chữ số): ");  String phoneNumber = scanner.nextLine();  // Kiểm tra số điện thoại hợp lệ  if (!isValidPhoneNumber(phoneNumber)) {  System.out.println("Số điện thoại không hợp lệ.");  return;  }  // Nếu mọi thứ hợp lệ  System.out.println("Thông tin hợp lệ!");  }  } |

Bài 2

|  |
| --- |
| import java.security.MessageDigest;  import java.security.NoSuchAlgorithmException;  import java.util.Scanner;  public class PasswordHashing {  // Hàm để băm mật khẩu bằng thuật toán SHA-256  public static String hashPassword(String password) {  try {  // Tạo một đối tượng MessageDigest với thuật toán SHA-256  MessageDigest digest = MessageDigest.getInstance("SHA-256");    // Băm chuỗi mật khẩu thành dạng byte  byte[] hash = digest.digest(password.getBytes());    // Chuyển giá trị băm dạng byte sang chuỗi dạng hex (mã thập lục phân)  StringBuilder hexString = new StringBuilder();  for (byte b : hash) {  // Chuyển từng byte sang dạng hex và thêm vào chuỗi  String hex = Integer.toHexString(0xff & b);  if (hex.length() == 1) {  hexString.append('0');  }  hexString.append(hex);  }    // Trả về chuỗi đã băm dạng hex  return hexString.toString();  } catch (NoSuchAlgorithmException e) {  throw new RuntimeException(e);  }  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Nhập mật khẩu từ người dùng  System.out.print("Nhập mật khẩu: ");  String password = scanner.nextLine();  // Gọi hàm để băm mật khẩu  String hashedPassword = hashPassword(password);  // In ra giá trị băm  System.out.println("Giá trị băm của mật khẩu (SHA-256): " + hashedPassword);  }  } |

Mật khẩu: password123

Giá trị băm (SHA-256):

ị:ef92b778bafe771e89245b89ecbcf97d7a09b653409d1a0e83f93efdc09e4ef8

Bài 3

|  |
| --- |
| +-------------------------------+  | Nhập thông điệp và khóa mã hóa |  +-------------------------------+  |  +--------------------------+  | Mã hóa thông điệp bằng AES |  +--------------------------+  |  +--------------------------+  | Lưu trữ thông điệp đã mã hóa |  +--------------------------+  |  +--------------------------+  | Giải mã thông điệp bằng AES |  +--------------------------+  |  +--------------------------+  | Hiển thị thông điệp gốc |  +--------------------------+ |

|  |
| --- |
| import javax.crypto.Cipher;  import javax.crypto.KeyGenerator;  import javax.crypto.SecretKey;  import javax.crypto.spec.SecretKeySpec;  import java.util.Base64;  import java.util.Scanner;  public class AESExample {  // Hàm để mã hóa thông điệp bằng AES  public static String encrypt(String message, SecretKey secretKey) throws Exception {  // Tạo đối tượng Cipher với thuật toán AES  Cipher cipher = Cipher.getInstance("AES");    // Khởi tạo Cipher ở chế độ mã hóa  cipher.init(Cipher.ENCRYPT\_MODE, secretKey);    // Mã hóa thông điệp  byte[] encryptedBytes = cipher.doFinal(message.getBytes());    // Trả về chuỗi đã mã hóa dưới dạng Base64  return Base64.getEncoder().encodeToString(encryptedBytes);  }  // Hàm để giải mã thông điệp đã mã hóa bằng AES  public static String decrypt(String encryptedMessage, SecretKey secretKey) throws Exception {  // Tạo đối tượng Cipher với thuật toán AES  Cipher cipher = Cipher.getInstance("AES");    // Khởi tạo Cipher ở chế độ giải mã  cipher.init(Cipher.DECRYPT\_MODE, secretKey);    // Giải mã thông điệp  byte[] decodedBytes = Base64.getDecoder().decode(encryptedMessage);  byte[] decryptedBytes = cipher.doFinal(decodedBytes);    // Trả về thông điệp đã giải mã  return new String(decryptedBytes);  }  public static void main(String[] args) {  try {  Scanner scanner = new Scanner(System.in);    // Nhập thông điệp từ người dùng  System.out.print("Nhập thông điệp cần mã hóa: ");  String message = scanner.nextLine();    // Sinh khóa bí mật AES (độ dài 128-bit)  KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");  keyGenerator.init(128); // AES sử dụng khóa 128-bit  SecretKey secretKey = keyGenerator.generateKey();    // Mã hóa thông điệp  String encryptedMessage = encrypt(message, secretKey);  System.out.println("Thông điệp đã mã hóa: " + encryptedMessage);    // Giải mã thông điệp  String decryptedMessage = decrypt(encryptedMessage, secretKey);  System.out.println("Thông điệp sau khi giải mã: " + decryptedMessage);    } catch (Exception e) {  e.printStackTrace();  }  }  } |

Thông điệp ban đầu: Hello, AES!

Thông điệp đã mã hóa: qLwDxgmsLpOrlMT6bkqrOQ==

Thông điệp sau khi giải mã: Hello, AES!

Bài 4

|  |
| --- |
| +---------------------------+  | Tạo cặp khóa RSA (công khai, bí mật) |  +---------------------------+  |  +-------------------+  | Nhập tài liệu cần ký |  +-------------------+  |  +--------------------+  | Tạo chữ ký số bằng RSA |  +--------------------+  |  +--------------------+  | Lưu chữ ký và tài liệu |  +--------------------+  |  +-------------------+  | Xác thực chữ ký số |  +-------------------+  |  +--------------------------+  | Chữ ký hợp lệ? |  +---------+----------------+  | Không | Có  +--------v-------+ +-------v--------+  | Thông báo thất bại | | Thông báo thành công |  +-------------------+ +--------------------+ |

|  |
| --- |
| import java.security.\*;  import java.util.Base64;  import javax.crypto.Cipher;  public class DigitalSignatureExample {  // Hàm để tạo cặp khóa RSA (gồm khóa công khai và khóa riêng)  public static KeyPair generateRSAKeyPair() throws NoSuchAlgorithmException {  KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");  keyGen.initialize(2048); // Sử dụng RSA 2048-bit  return keyGen.generateKeyPair();  }  // Hàm để ký tài liệu (chữ ký số) bằng khóa riêng  public static String signDocument(String document, PrivateKey privateKey) throws Exception {  Signature rsa = Signature.getInstance("SHA256withRSA");  rsa.initSign(privateKey);  rsa.update(document.getBytes());    byte[] signature = rsa.sign();  return Base64.getEncoder().encodeToString(signature); // Trả về chữ ký đã mã hóa dạng Base64  }  // Hàm để xác thực chữ ký bằng khóa công khai  public static boolean verifySignature(String document, String signature, PublicKey publicKey) throws Exception {  Signature rsa = Signature.getInstance("SHA256withRSA");  rsa.initVerify(publicKey);  rsa.update(document.getBytes());    byte[] signatureBytes = Base64.getDecoder().decode(signature);  return rsa.verify(signatureBytes); // Trả về kết quả xác thực (true/false)  }  public static void main(String[] args) {  try {  // Tạo cặp khóa RSA  KeyPair keyPair = generateRSAKeyPair();  PublicKey publicKey = keyPair.getPublic();  PrivateKey privateKey = keyPair.getPrivate();  // Tài liệu cần ký  String document = "Đây là tài liệu cần được ký số.";  System.out.println("Tài liệu gốc: " + document);  // Tạo chữ ký số bằng khóa riêng  String signature = signDocument(document, privateKey);  System.out.println("Chữ ký số: " + signature);  // Xác thực chữ ký bằng khóa công khai  boolean isVerified = verifySignature(document, signature, publicKey);  if (isVerified) {  System.out.println("Xác thực thành công. Tài liệu không bị chỉnh sửa.");  } else {  System.out.println("Xác thực thất bại. Tài liệu có thể đã bị chỉnh sửa.");  }  } catch (Exception e) {  e.printStackTrace();  }  }  } |

Tài liệu gốc: Đây là tài liệu cần được ký số.

Chữ ký số: mSYvZcKNBqflm7dCM8vPV/+WVp...

Kết quả xác thực: Xác thực thành công. Tài liệu không bị chỉnh sửa.

Bài 5

|  |
| --- |
| +--------------------------+  | Người dùng truy cập trang |  +--------------------------+  |  +---------------------------+  | Sinh CSRF Token và hiển thị |  +---------------------------+  |  +---------------------------+  | Nhập thông tin đăng nhập và |  | CSRF Token gửi đến server |  +---------------------------+  |  +---------------------------+  | Kiểm tra CSRF Token hợp lệ |  +---------------------------+  / \  +/------+ +-------+\  | Hợp lệ | | Không |  | | | Hợp lệ |  +--------+ +--------+  | |  +------------------------+ |  | Kiểm tra đăng nhập với | |  | Prepared Statement (SQL)| |  +------------------------+ |  | |  +--------------------+ |  | Đăng nhập thành công | |  +--------------------+ |  +---v---+  | Thông báo |  | lỗi |  +----------+ |

SQL:

|  |
| --- |
| CREATE TABLE users (  id INT PRIMARY KEY AUTO\_INCREMENT,  username VARCHAR(50) NOT NULL,  password VARCHAR(50) NOT NULL  );  INSERT INTO users (username, password) VALUES ('admin', 'admin123'); |

Java:

|  |
| --- |
| import java.security.SecureRandom;  import java.sql.\*;  import java.util.Base64;  import java.util.Scanner;  public class CSRFAndSQLInjectionProtection {  // Hàm để sinh CSRF Token ngẫu nhiên  public static String generateCSRFToken() {  SecureRandom secureRandom = new SecureRandom();  byte[] token = new byte[16]; // Token có độ dài 16 byte  secureRandom.nextBytes(token);  return Base64.getEncoder().encodeToString(token); // Mã hóa thành Base64 để dễ lưu trữ  }  // Hàm kiểm tra thông tin đăng nhập sử dụng Prepared Statement để chống SQL Injection  public static boolean validateLogin(String username, String password, Connection connection) throws SQLException {  String query = "SELECT \* FROM users WHERE username = ? AND password = ?";  PreparedStatement statement = connection.prepareStatement(query);  statement.setString(1, username); // Gán tham số 1 là username  statement.setString(2, password); // Gán tham số 2 là password  ResultSet resultSet = statement.executeQuery();  return resultSet.next(); // Trả về true nếu có bản ghi phù hợp  }  // Hàm giả lập việc kiểm tra CSRF Token  public static boolean validateCSRFToken(String receivedToken, String sessionToken) {  return receivedToken.equals(sessionToken);  }  public static void main(String[] args) {  try {  // Kết nối tới cơ sở dữ liệu MySQL  Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "password");  // Giả lập sinh CSRF Token khi người dùng truy cập trang  String sessionCSRFToken = generateCSRFToken();  System.out.println("CSRF Token được sinh ra: " + sessionCSRFToken);  // Giả lập form đăng nhập  Scanner scanner = new Scanner(System.in);  System.out.print("Nhập tên đăng nhập: ");  String username = scanner.nextLine();  System.out.print("Nhập mật khẩu: ");  String password = scanner.nextLine();  System.out.print("Nhập CSRF Token: ");  String receivedCSRFToken = scanner.nextLine();  // Kiểm tra tính hợp lệ của CSRF Token  if (!validateCSRFToken(receivedCSRFToken, sessionCSRFToken)) {  System.out.println("CSRF Token không hợp lệ. Yêu cầu bị từ chối.");  return;  }  // Kiểm tra thông tin đăng nhập với Prepared Statement  if (validateLogin(username, password, connection)) {  System.out.println("Đăng nhập thành công!");  } else {  System.out.println("Tên đăng nhập hoặc mật khẩu không chính xác.");  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  }  } |

CSRF Token được sinh ra: WXYgMwXHevYzTxMfzAj4ZA==

Nhập tên đăng nhập: admin

Nhập mật khẩu: admin123

Nhập CSRF Token: WXYgMwXHevYzTxMfzAj4ZA==

Kết quả: Đăng nhập thành công!