**Info about this Java Project**

Prompt Start

I am an undergraduate student that studies information technology. I do not know much about programming and would like to make a Java project to help build my knowledge. Can you be an assistant to help me make a project. I will provide more information in later prompts. Also clarify throughout the process if what was outputted okay. Just make a small short response.

Context

I am nearly going to graduate and need to do 1 more advanced IT subject. I was advised to take a subject 1 semester as I couldn’t take a leave of absence.

*“If you pass your course in semester 2, 2024, you will be left with 1 more advanced IT option course. You can choose either of*

*Cloud Security or Blockchain fundamentals for this last course. Here are the course guides:*

* *Cloud Security:* [*http://www1.rmit.edu.au/courses/038407*](http://www1.rmit.edu.au/courses/038407)
* *Blockchain fundamentals:* [*http://www1.rmit.edu.au/courses/INTE26272450*](http://www1.rmit.edu.au/courses/INTE26272450)

*You mentioned that you want to take LoA for semester 1, because you want to take System Deployment & Operations (COSC2759).”*

* *Shekhar Kalra*

Here is the course information for Cloud Security, one of the courses I need to do enclosed in brackets:

{

**Course Title:**Cloud Security

**Credit Points:**12.00

**Course Description**

Cloud security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of technologies deployed to protect data, applications, and the associated infrastructure of cloud computing.

This course provides a practical survey of both the principles and practice of cloud security. The emphasis of this course is on the underlying principles and techniques of cloud security with examples of how they are applied in practice.

Please note that if you take this course for a bachelor honours program, your overall mark in this course will be one of the course marks that will be used to calculate the weighted average mark (WAM) that will determine your award level. This applies to students who commence enrolment in a bachelor honours program from 1 January 2016 onwards.

**Program Learning Outcomes**

This course is an option course and not a core course, so that it does not need to contribute to the Program Learning Outcomes.

**Course Learning Outcomes**

Upon successful completion of this course you should be able to:

1. Demonstrate knowledge of cloud security principles and mechanisms
2. Demonstrate computer programming and configuration skills required to develop a cloud security infrastructure
3. Identify cloud security weaknesses by recognising and discovering threats and vulnerabilities to cloud computing
4. Problem solve how to fix cloud security weaknesses and mitigate security threats to cloud computing
5. Demonstrate knowledge and skills to prepare for industry cloud security certificate exams e.g. CCSK, CCSP
6. Communicate clearly and effectively using the technical language of the field correctly

**Overview of Learning Activities**

The learning activities included in this course are:

* Lectorials: Key concepts will be explained in lectorials in which course material will be presented and the subject matter will be illustrated with demonstrations and examples;
* Tutorial sessions focus on analysing and problem solving of given scenarios, such as security analysis and secure solutions;
* Computer laboratory sessions provide practices in the application of security techniques.

}

Here is information from one of tutors about what I should to prepare for the subject enclosed in brackets.

{

The programming tasks for this course are expected to be completed in Java or JavaScript. Generally speaking, and without going into too much detail, we use a programming language to apply a cryptographic concept, where the objective is to transform data into a form that appears random, but the original data can be recovered.

As a simplified example of this idea, consider that we have a character string like "Hello World!" and the task is to transform each character, such that the transformation can be reversed. For this task, the programming constructs we would need to use are:

* **String handling**: obtaining individual characters.
* **Basic data types**: understanding the relationship between character and integer data types.
* **IF statement:** can be used to handle characters like '!' and the space character as special cases.
* **FOR loop:** need a way to iterate over all characters in a string.
* **Arithmetic:** need to use a mathematical operation that can be reversed.

These constructs are found in either Java or JavaScript. In the case of JavaScript, we also need an API that allows us to get input from a webpage to make the implementation more interactive (that is, the Document Object Model that represents the HTML document). We typically avoid the use of language features like polymorphism or exception handling, because this can distract from the task at hand. For more advanced algorithms, we make use of programming libraries, which are provided as needed.

}

***Keep in mind I am unfamiliar in Java and I would like some projects that cover the basics before proceeding to difficulty projects. Rank the projects also in difficulties.***

# List of projects

**Beginner-Level Projects**

1. **Character Transformation Program**  
   *Difficulty: Easy*
   * **Goal**: Write a program that takes a string as input, applies a reversible transformation to each character (e.g., shift each character by a fixed number using ASCII values), and then reverses the transformation.
   * **Skills Covered**:
     + String handling
     + For loops and conditionals
     + Basic data types (e.g., char and int)
     + Arithmetic operations
2. **Basic File Encryption/Decryption**  
   *Difficulty: Easy*
   * **Goal**: Create a program that reads a text file, applies a simple encryption method (e.g., Caesar cipher), and writes the encrypted output to another file. Include a decryption option.
   * **Skills Covered**:
     + File handling
     + String manipulation
     + Loops and conditionals

**Intermediate-Level Projects**

1. **Password Hashing Simulator**  
   *Difficulty: Moderate*
   * **Goal**: Build a program that simulates password hashing using a cryptographic hash function like SHA-256 (use Java’s built-in libraries for hashing).
   * **Skills Covered**:
     + Using external libraries (java.security package)
     + Reading user input
     + Understanding hash algorithms
2. **Symmetric Encryption with a Key**  
   *Difficulty: Moderate*
   * **Goal**: Implement a basic symmetric encryption algorithm (e.g., AES). The program should encrypt and decrypt text using a shared key provided by the user.
   * **Skills Covered**:
     + Cryptographic libraries (e.g., javax.crypto)
     + Working with byte arrays and strings
     + Exception handling (basic)

**Advanced-Level Projects**

1. **Secure Cloud File Storage Emulator**  
   *Difficulty: Hard*
   * **Goal**: Create a simple program to encrypt a file before uploading it to a simulated cloud directory (use local storage as the "cloud"). Include options to decrypt and download the file.
   * **Skills Covered**:
     + File I/O
     + Encryption and decryption (e.g., AES)
     + Basic command-line interface
2. **Authentication and Token Generation System**  
   *Difficulty: Hard*
   * **Goal**: Build a basic authentication system where users enter their credentials, and the program generates a token (e.g., JWT). This token can then be validated.
   * **Skills Covered**:
     + String encoding/decoding (e.g., Base64)
     + Token creation and validation
     + Libraries for cryptographic functions

**Progression Plan**

1. Start with **Character Transformation Program** (1-2 weeks).
2. Move to **Basic File Encryption/Decryption** (2 weeks).
3. Transition to intermediate projects like **Password Hashing Simulator** (2 weeks).
4. Build your skills with **Symmetric Encryption with a Key** (2 weeks).
5. End with one of the advanced projects like **Secure Cloud File Storage Emulator** or **Authentication and Token Generation System** (4 weeks).

# Project 1

## Prompt

This definitely aligns with my goals. I will start with the Character Transformation Program. I will start the project today and have a duration of 2 weeks. Give me more details of the Character Transformation Program in Java so I understand what it does and the constraints.

## Details

**Objective**

The program will:

1. Take a string input from the user (e.g., "Hello World!").
2. Transform each character using a reversible operation (e.g., shifting characters by a specific number in the ASCII table).
3. Provide the option to reverse the transformation (i.e., decode the string back to its original form).

**Features**

1. **Input Handling**:
   * The user will input a string to be transformed.
   * Allow handling of special cases like spaces (' ') and punctuation ('!', '.', etc.).
2. **Transformation Logic**:
   * Apply a simple transformation to each character:  
     For example, shift each character by 3 positions in the ASCII table.
     + 'A' becomes 'D'
     + 'z' becomes 'c'
   * Ensure the transformation is reversible, so the program can decode the transformed string back to its original form.
3. **Constraints**:
   * Handle both uppercase and lowercase letters.
   * Keep non-alphabetic characters (e.g., spaces, punctuation) unchanged.
   * Use modular arithmetic to wrap around the alphabet. For example:
     + 'z' shifted by 3 wraps back to 'c'.
4. **Menu Options**:
   * Provide a menu to the user:
     + Encode a string
     + Decode a string
     + Exit the program

**Constraints**

* The program should handle strings of any length.
* Assume the user will input valid text (no need to handle numbers or special cases for now).
* Focus on simplicity and clarity—no need to optimize for performance at this stage.

**Example**

**Input:**

1. The user enters the string "Hello World!" to encode.
2. Each character is shifted by 3 positions.
   * 'H' becomes 'K'
   * 'e' becomes 'h'
   * 'l' becomes 'o'
   * ' ', 'W', '!' remain unchanged.

**Output:**

**Encoded String**: "Khoor Zruog!"

1. The user selects the decode option and inputs "Khoor Zruog!".  
   **Decoded String**: "Hello World!"

**Structure**

Here’s how your program can be broken down:

1. **Main Class**: Contains the program entry point and menu logic.
2. **Methods**:
   * encode(String input, int shift)
   * decode(String input, int shift)
   * displayMenu()

## Tools and Deadline

Here’s a list of tools to help you develop and manage your project:

1. **Integrated Development Environment (IDE)**:
   * **Recommendation**: Use **Eclipse** or **IntelliJ IDEA (Community Edition)**.
   * Both are beginner-friendly and provide useful features like syntax highlighting, auto-completion, and debugging tools.
2. **Java Development Kit (JDK)**:
   * Install the **latest stable version of JDK** (e.g., JDK 17 or 21).
   * Make sure your IDE is configured to use the installed JDK.
3. **Compiler and Runtime**:
   * These come with the JDK, so you’ll use javac (Java Compiler) and java (Java Runtime) behind the scenes.
4. **Version Control (Optional but Recommended)**:
   * Use **Git** with a platform like **GitHub** or **GitLab** to track your progress.
5. **Text Editor (Optional)**:
   * Use a simple editor like **Notepad++** or **VS Code** for jotting down pseudocode or testing small snippets.
6. **Command Line or Terminal**:
   * Learn basic terminal commands to compile and run Java programs, as this deepens your understanding of the Java build process.

**Timeline**

Your project duration is 2 weeks. Here’s a suggested schedule:

**Week 1:**

* **Day 1-2**:
  + Set up your environment (IDE, JDK).
  + Familiarize yourself with basic Java syntax (data types, loops, conditionals).
  + Understand the ASCII table (important for character transformations).
  + Write a basic program to manipulate and print characters.
* **Day 3-5**:
  + Implement the **encode** function.
  + Test it with different strings (e.g., uppercase, lowercase, special characters).
* **Day 6-7**:
  + Implement the **decode** function.
  + Combine both into a single menu-driven program.

**Week 2:**

* **Day 8-10**:
  + Test edge cases (e.g., empty strings, long strings, only special characters).
  + Add comments and improve code readability.
* **Day 11-13**:
  + Refactor the program into separate methods for clarity.
  + Add user-friendly error messages (e.g., invalid menu input).
* **Day 14** (Final Day):
  + Conduct final tests.
  + Document your work with a simple **README** file explaining how to use the program.

**Completion Date**

* **Start Date**: November 18, 2024
* **End Date**: December 2, 2024

## How should the menu look like?

**How the Menu Should Work**

1. The program starts with a **welcome message** or title (e.g., "Welcome to the Character Transformation Program").
2. The user is shown a **menu with options** to select:
   * Option **1**: Encode a string.
   * Option **2**: Decode a string.
   * Option **3**: Exit the program.
3. The program **waits for the user's input** (e.g., 1, 2, or 3).
4. Based on the input:
   * If **1** is selected, the program asks the user to enter a string to encode and then displays the encoded result.
   * If **2** is selected, the program asks the user to enter an encoded string to decode and then displays the decoded result.
   * If **3** is selected, the program exits with a goodbye message.
5. If the user enters an **invalid option**, the program should notify them and display the menu again.

A screenshot of a computer program

Description automatically generated

## What is a module

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

# Know how to use classic tokens

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens>

* This is to replace passwords and make it more secure to commit changes. This is better as it allows read and write.

# Links

How to get started and create a Java File: <https://youtu.be/Cf2TFrZW2l0>

* + - This is shorter and better: https://www.youtube.com/watch?v=VDLgUo120\_I

How to add Eclipse to Git: https://www.youtube.com/watch?si=1ClVl322ltK6XHwu&v=LPT7v69guVY&feature=youtu.be

How to delete a repository: https://www.youtube.com/watch?v=u4qVRJOqMH0