1. 
2. 

### Student's Notes: Single Responsibility Principle

#### Introduction

* **Single Responsibility Principle (SRP)** is the first principle from the SOLID Design Principles group.
* **Formal Definition:** There should never be more than one reason for a class to change.
* **Simplified Meaning:** A class should have a very focused functionality or address a specific concern of the desired functionality.

#### Understanding the Principle

* A class with a single responsibility is easier to maintain and change.
* When a class has multiple responsibilities, changes in one responsibility can affect others, leading to complex modifications.

#### Example to Understand SRP (Single Responsibility Principle)

1. **Scenario:** A class creates and sends a message to a remote server.
2. **Potential Reasons for Change:**
   * Change in the communication protocol.
   * Change in the message format (e.g., from JSON to XML).
   * Addition of new parameters for communication, such as authentication.
3. **Problem:** If all responsibilities are in a single class, any of the above changes will require modifications to the same class, making it harder to maintain and prone to errors.

#### How to Follow SRP

* Divide responsibilities into separate classes or modules:
  + Example: One class for creating the message, another for sending it, and another for handling authentication.
* Benefits:
  + Each class addresses a specific concern.
  + Changes are isolated to the relevant class, making code more organized and maintainable.

#### Key Takeaways

* **Single Responsibility:** A class should address only one concern.
* When a class has multiple responsibilities, split it into separate classes or modules.
* This principle ensures that a class has only one reason to change, reducing the likelihood of introducing errors during modifications.

#### Practical Application

* When designing a class or module, ensure that it is focused on a single functionality.
* When changes occur, modify only the specific class responsible for that functionality, avoiding **ripple effects** on unrelated parts of the code.

#### Reflection on the Definition

* The definition, "There should never be more than one reason for a class to change," becomes clearer when you understand that it aims to simplify code maintenance and reduce **interdependencies**.

#### Next Steps

* Apply SRP in code by creating separate classes for distinct functionalities.
* Observe how following SRP makes your code easier to manage and modify.