Single Responsibility Principle

Each class should be responsible for only one task and should have no other reason change. This makes the code easily understand,modify and test.

Let’s consider an example of a class name **Customer** which is responsible for calculating the final bill of a customer.

It has several responsibilities such as maintaining the customer details, calculating the final bill, and generating an invoice.

```
Class CustomerDetails
{
  private:
     string name;
      int Id;
      vector<Item> items;
      float totalAmount;
   public:
      string getName();
      void setName(string name);
      void setId(int id);
       int getId();
       void addItem(Item item);
       void removeItem(Item item);
       float calculatorTotalAmount();
       string generateInvoice();
 };
```

As we can see, the **Customer** class is responsible for multiple tasks, such as maintaining customer details, calculating the final bill, and generating an invoice. This violates the Single Responsibility Principle, as the class has multiple reasons to change. For instance, if the calculation of the final bill is changed, then the **Customer** class needs to be modified, even though it is not related to the customer details or the generation of invoices.

To implement SRP, we can create separate classes for each of these responsibilities. For example, we can create a **CustomerDetails** class to handle the customer details, a **BillingCalculator** class to calculate the final bill, and an **InvoiceGenerator** class to generate the invoice.

Class **CutomerDetails**

```
{
  private:
     string name;
     int Id;
  public:
      void setName(String name);
       void setId(int Id);
       int getId();
       string getName();
 };
```

class **BillingCalculator**

```
{
   private:
      vector<Item> items;
   public:
      void addItem(Item item);
       void removeItem(Item item);
        float calculateTotalAmount();
};
```

class **InvoiceGenerator**

```
{
    public:
        string generateInvoice( CustomerDetails m_customerDetails, BillingCalculator m_billCalculator);
};
```

By dividing the responsibilities of the **Customer** class into smaller, more focused classes, each class has only one responsibility, which makes the code more maintainable and scalable. If any changes are required in any of these classes, only that class needs to be modified, rather than the entire **Customer** class.