Abstraction

The concept of abstraction is to reduce something down to its most basic or simple components, with the goal of making something simpler to read, understand, and use. This also often means displaying to the user a minimal amount of information as well as encapsulating large categories or objects to hide its complexity. In programming, this often means that each piece of the program accomplishes a single job, each variable holds a single value, each object holds only necessary information, and each class only holds necessary objects. This makes it much easier to write and retrieve both the code and the contents of an application, as well as ensuring that information follows a standard format.  
To share an example, let me display the following code segment:  
do

    {

      Console.WriteLine("\nPlease select one of the following options:");

      Console.WriteLine("1. Read past Journal Entries");

      Console.WriteLine("2. Write a new Journal Entry");

      Console.WriteLine("3. Change file location");

      Console.WriteLine("4. Quit");

      Console.Write("\nWhat would you like to do? ");

      int choice;

      while (!int.TryParse(Console.ReadLine(), out choice) || choice < 1 || choice > 4)

      {

        Console.WriteLine("Invalid choice. Please enter a number between 1 and 4.");

      }

      Console.WriteLine();

      // Switch statement handles user input, exceeds requirements with validation

      switch (choice)

      {

        case 1:

          journal.DisplayJournal();

          break;

        case 2:

          journal.WriteJournal();

          break;

        case 3:

          // Default location is bin/Debug/net6.0/journal.csv

          // This csv file is excel compatible and exceeds requirements

          Console.WriteLine("Please enter the file path for your journal (e.g., C:\\path\\to\\your\\journal.csv): ");

          journal.SetFilePath(Console.ReadLine());

          break;

        case 4:

          Console.WriteLine("Thank you for using the Journal Program. Goodbye!");

          repeat = false;

          break;

      }

    } while (repeat);

In this code segment, the program is writing a menu for the user to interact with. Instead of listing everything haphazardly, each line displayed to the user regarding their choices is distinct from one another, making it easier for the user to read. Then, after the user enters a number, the menu checks that number against its cases, with each case accomplishing one task. Now, if I wanted to edit a task, add or move menu options, or change anything else about this program, I can easily identify what needs to be changed and I can do so with minimal line edits, only changing the relevant pieces instead of altering a lot of code.