**Abstractions**

Abstraction is basically mean hiding complex part of a computer program and exposing only the necessary parts to the user of the program. It allows programmers to reduce and factor out details so that they can focus on a few concepts at a time. In essence, abstraction helps to manage complexity by providing a simplified model of a more complex system. A practical way of visualizing the concept of Abstraction is to consider the television set. The operator of a Television set is only interested in how to control the Television set externally such as: turning on/off, connecting other devices to it among others. The code behind these functions are hidden from the user. These various functions are broken down in to smaller and simple code for easy comprehension.

Abstraction is crucial because it simplifies problem-solving by allowing developers to break down complex systems into manageable parts. This makes it easier to understand, develop, and maintain software. By focusing on higher-level concepts, programmers can work more efficiently and effectively without getting bogged down by intricate details.

In the code below, demonstrate some implementation of abstractions.

In the Resume class, jobs are stored in a List<Job> \_jobs. This collection abstracts the individual Job instances, allowing the Resume class to manage and display multiple jobs without knowing or caring about the internal workings of each Job object.

using System;

using System.Collections.Generic;

public class Resume

{

// Member variables

public string \_name;

**public List<Job> \_jobs = new List<Job>();**

// Method to display resume details

public void Display()

{

Console.WriteLine($"Name: {\_name}");

Console.WriteLine("Jobs:");

foreach (var job in \_jobs)

{

job.Display();

}

}

}