**What is Inheritance and Why is it Important?**

Inheritance is the third principle of programming with classes. To understand this principle it is important to remember that inheritance is a mechanism for code reuse. Instead of writing repeating code, a child class can just inherit these from a parent class. Inheritance is when one class obtains the attributes and methods of another class directly, without having to type them. Just like people inherit certain characteristics from their parents.

For example, a Person and a Student. A person can have attributes and methods that all people share, such as GetName(). Because a student is a person, a student should have all the properties and behaviors that a person does, However a student could have more specific information, such as a student ID number, which could be accessed via a GetNumber() method.

**Benefit of inheritance**

One significant benefit of inheritance is that makes it possible to create a hierarchy of classes. By putting common variables and methods in a base class, code duplication can be avoided. For example, if a program has a Person class with shared methods like GetName(), a Student class can inherit from the Person class and gain access to these methods while also introducing its own unique features, such as GetNumber().

**Application of Inheritance**

An application of encapsulation can be seen in my assignment management program, where the “WritingAssignment” class inherits from the Assignment class. This makes it possible for the “WritingAssignment” to use common properties and methods from the Assignment class while also introducing its own unique features. This structure is reusable and logical, which makes it easier to manage different types of assignments.

**Code Example: Inheritance**

public class Program

{

static void Main(string[] args)

{

Assignment a1 = new Assignment("Samuel Bennett", "Multiplication");

Console.WriteLine(a1.getSummary());

Console.WriteLine(" ");

MathAssignment mathAssignment1 = new MathAssignment("Roberto Rodriguez", "Fractions", "7.3", "8-19");

Console.WriteLine(mathAssignment1.getSummary());

Console.WriteLine(mathAssignment1.GetHomeworkList());

Console.WriteLine(" ");

WritingAssignment writingAssignment1 = new WritingAssignment("Mary Waters", "European History", "The Causes of World War II");

Console.WriteLine(writingAssignment1.getSummary());

Console.WriteLine(writingAssignment1.GetWritingInformation());

}

}

public class Assignment

{

protected string \_studentName;

protected string \_topic;

public Assignment(string studentName, string topic)

{

\_studentName = studentName;

\_topic = topic;

}

public string getSummary()

{

return $"student: {\_studentName}, topic: {\_topic}";

}

}

public class MathAssignment : Assignment

{

private string \_textbookSection;

private string \_problems;

public MathAssignment(string studentName, string topic, string textbookSection, string problems)

: base(studentName, topic)

{

\_textbookSection = textbookSection;

\_problems = problems;

}

public string GetHomeworkList()

{

return $"Section: {\_textbookSection}, Problems: {\_problems}";

}

}

public class WritingAssignment : Assignment

{

private string \_title;

public WritingAssignment(string studentName, string topic, string title)

: base(studentName, topic)

{

\_title = title;

}

public string GetWritingInformation()

{

return $"Title: {\_title}, Author: {\_studentName}";

}

}

**Explanation**

In this example, the Assignment class is the base class, encapsulating shared properties like \_studentName and \_topic. The MathAssignment and WritingAssignment classes inherit these properties from Assignment, gaining access to these properties and methods.

MathAssignment has specific attributes related to math assignments, such as \_textbookSection and \_problems, and a method to display the homework list.

Similarly, WritingAssignment has specific attributes related to writing assignments, \_title, and a method to present writing information.