CHEATSHEET - Polymorphism + Adv OOP + Text File

POLYMORPHISM + ADV OOP + TEXT FILE

Polymorphic Parameters

Concept: Polymorphism allows methods to accept parameters of different types through base class or interface references.

Example :

```
public void DisplayShape(Shape shape) {
  shape.Draw();
}
```

Sealed Keyword

Concept: Prevents a class from being inherited or a method from being overridden.

Syntax :

```
sealed class SealedClass {
  // Members
}

class BaseClass {
  public sealed void SealedMethod() {
    // Implementation
  }
}
```

tutorials.EU

Has A - Relationships

Concept: Demonstrates composition, where a class contains instances of other classes to use their functionality.

Example :

```
class Engine {
  // Engine properties and methods
}

class Car {
  private Engine engine = new Engine();
  // Car properties and methods
}
```

Abstract

Concept: Abstract classes provide a base class with abstract methods that must be implemented by derived classes.

Syntax :

```
abstract class Animal {
  public abstract void MakeSound();
}

class Dog : Animal {
  public override void MakeSound() {
    Console.WriteLine("Bark");
  }
}
```

as & is Keyword / Polymorphism

Concept: Using abstract classes with as and is keywords for type checking and casting in polymorphic scenarios

Example :

```
if (animal is Dog) {
  Dog dog = animal as Dog;
  dog.Bark();
}
```

Interfaces vs Abstract Classes

Comparison:
Interfaces: Define a contract without implementation.
Abstract Classes: Can provide both a contract and some implementation.

Syntax :

```
interface IAnimal {
  void MakeSound();
}

abstract class Animal {
  public abstract void MakeSound();
}
```

File Operations in C#

Read from a Textfile
Concept: Reading data from a text file using StreamReader.

Example :

```
using (StreamReader sr = new
StreamReader("file.txt")) {
  string content = sr.ReadToEnd();
}
```

Write into a Text File
Concept: Reading data from a text file using StreamReader.

Example :

```
using (StreamWriter sw = new
StreamWriter("file.txt")) {
  sw.WriteLine("Hello, World!");
}
```

Summary

Polymorphism: Allows methods to operate on objects of different types through base class or interface references.
Sealed Keyword: Prevents further inheritance or method overriding.
Has A - Relationships: Demonstrates composition where a class contains instances of other classes.
Abstract Classes: Provide a template for derived classes with mandatory method implementations.
Interfaces vs Abstract Classes: Understanding when to use interfaces versus abstract classes.
File Operations: Techniques for reading from and writing to text files in C#.