

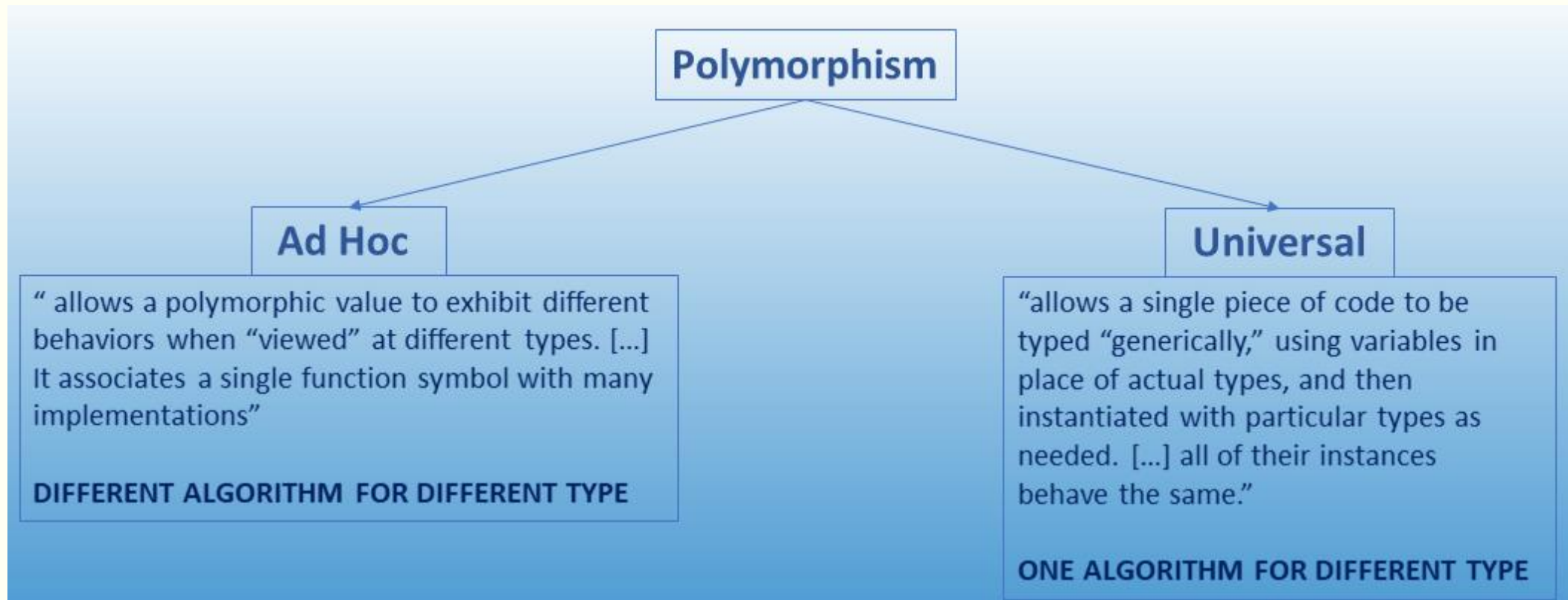


ПОЛІМОРФІЗМ ТА ЙОГО РІЗНОВИДИ

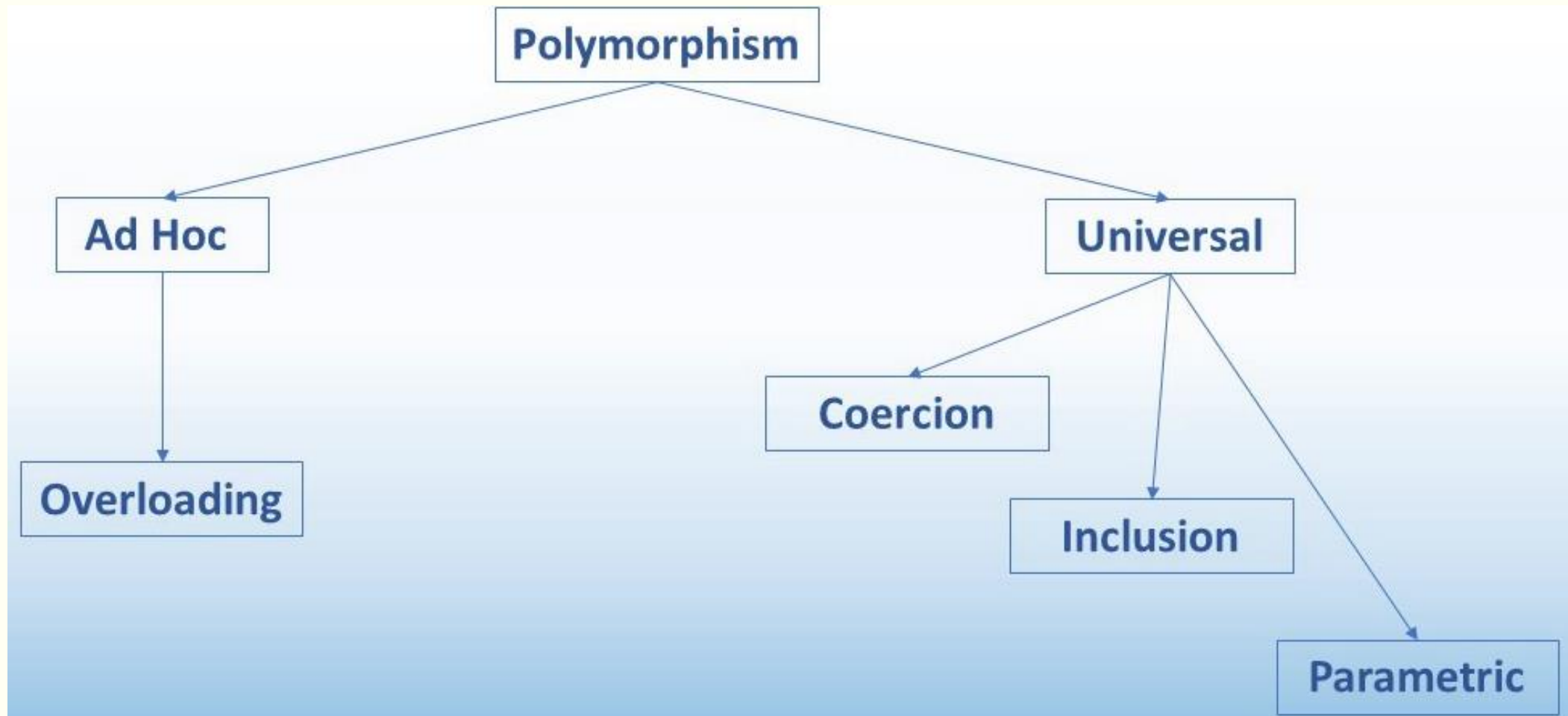
Питання 4.3.

Поняття та різновиди поліморфізму

- Термин “поліморфізм” означає семейство різних механізмів, позволяючих використовувати один і той же участок програми з різними типами в різних контекстах.

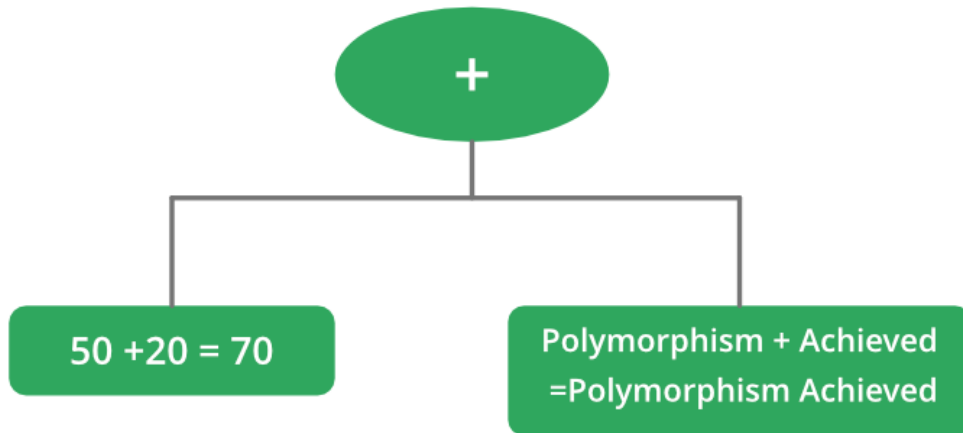


Поняття та різновиди поліморфізму



Спеціалізований (Ad-hoc) поліморфізм

Ad-hoc Polymorphism



- Ad-hoc поліморфізм також називають *перевантаженням (Overloading)*.
 - Він дозволяє функціям з однаковою назвою працювати по-різному.
 - Наприклад:
Оператор + додає 2 цілих числа та виконує конкатенацію рядків.
- Перевантаження методів – поширений спосіб впровадження поліморфізму.
 - Реалізація перевантаження полягає в означенні 2 або більше методів у класі з одною назвою.
 - С# може розрізняти методи з різними сигнатурами, проте однією назвою.
- Навіщо перевантаження методів?
 - Наприклад, для реалізації однієї операції для різних вхідних даних.

Перевантаження (overloading) методів у C#

- Перевантажені методи відрізняються на основі кількості та типів параметрів, переданих у методи.
- Не можна визначати більше одного методу з одною назвою, порядком та типом аргументів. Це викличе помилку компіляції.
- Компілятор не розглядає вихідний тип (return type) при перевантаженні методів.
 - Проте оголосити 2 методи з однаковою сигнатурою, але різними вихідними типами неможливо.
- Перевантаження методів може виконуватись шляхом зміни:
 - Кількості параметрів у 2 методах.
 - Типів даних параметрів методів.
 - Порядку параметрів методів.

Перевантаження (overloading) методів у C#

```
public void Identity(String name, int id) {  
    Console.WriteLine("Name : " + name + ", " + "Id : " + id);  
}  
  
// додавання 2 цілих чисел.  
public int Add(int a, int b) { return a + b; }  
  
// зміна кількості параметрів:  
// додавання трьох цілих чисел.  
public int Add(int a, int b, int c) {  
    return a + b + c;  
}  
  
// зміна типів параметрів:  
// додавання трьох дійсних чисел подвійної точності.  
public double Add(double a, double b, double c) {  
    double sum = a + b + c;  
    return sum;  
}  
  
// зміна порядку параметрів  
public void Identity(int id, String name) {  
    Console.WriteLine("Name : " + name + ", " + "Id : " + id);  
}
```

Перевантаження (overloading) методів у C#

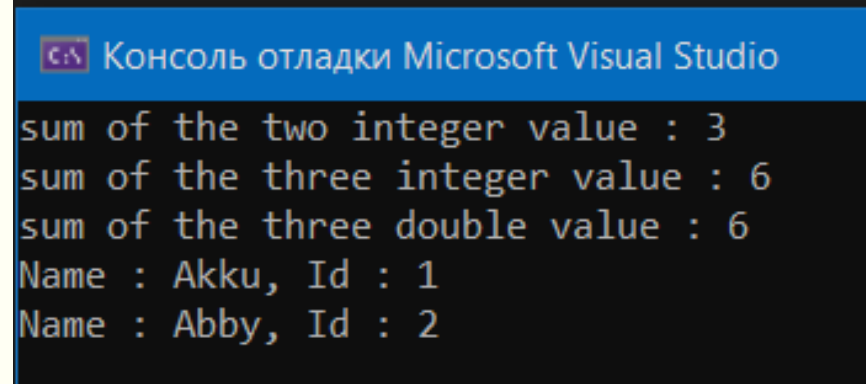
```
static void Main(string[] args)
{
    Program obj = new Program();
    int sum1 = obj.Add(1, 2);

    Console.WriteLine("sum of the two "
        + "integer value : " + sum1);

    int sum2 = obj.Add(1, 2, 3);
    Console.WriteLine("sum of the three "
        + "integer value : " + sum2);

    double sum3 = obj.Add(1.0, 2.0, 3.0);
    Console.WriteLine("sum of the three "
        + "double value : " + sum3);

    obj.Identity("Akku", 1);
    obj.Identity("Abby", 2);
}
```



Консоль отладки Microsoft Visual Studio

```
sum of the two integer value : 3
sum of the three integer value : 6
sum of the three double value : 6
Name : Akku, Id : 1
Name : Abby, Id : 2
```

Перевантаження операторів

- Перевантаження операторів дає можливість використовувати один оператор для різних операцій.
 - Доповнює функціональність операторів C# при їх застосуванні до користувацьких типів даних.
 - Перевантажити можна лише заздалегідь заданий набір операторів.
- The function of the operator is declared by using the **operator** keyword.

```
модифікаторДоступу НазваКласу operator символОператора (параметри)
{
    // код
}
```


Перевантаження унарних операторів

```
class Calculator
{
    public int number1, number2;
    public Calculator(int num1, int num2)
    {
        number1 = num1;
        number2 = num2;
    }

    // функція для виконання операції
    // зміни знаку цілих чисел
    public static Calculator operator -(Calculator c1)
    {
        c1.number1 = -c1.number1;
        c1.number2 = -c1.number2;
        return c1;
    }

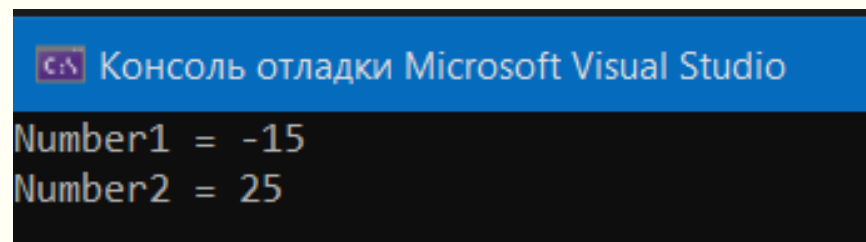
    // метод для виводу чисел
    public void Print()
    {
        Console.WriteLine("Number1 = " + number1);
        Console.WriteLine("Number2 = " + number2);
    }
}
```

04.10.2020

```
static void Main(string[] args)
{
    // застосовуємо перевантажений оператор -
    // з об'єктом класу
    Calculator calc = new Calculator(15, -25);

    calc = -calc;

    // виводимо результат
    calc.Print();
}
```



Консоль отладки Microsoft Visual Studio

```
Number1 = -15
Number2 = 25
```

Перевантаження бінарних операторів

```
class Calculator2
{
    public int number = 0;

    // безаргументний конструктор
    public Calculator2() { }

    // параметризований конструктор
    public Calculator2(int n)
    {
        number = n;
    }

    // перевантаження бінарного оператора "+"
    public static Calculator2 operator +(Calculator2 Calc1,
                                       Calculator2 Calc2)
    {
        Calculator2 Calc3 = new Calculator2(0);
        Calc3.number = Calc2.number + Calc1.number;
        return Calc3;
    }

    // метод для виводу результату
    public void display()
    {
        Console.WriteLine("{0}", number);
    }
}
```

```
static void Main(string[] args)
{
    Calculator2 num1 = new Calculator2(200);
    Calculator2 num2 = new Calculator2(40);
    Calculator2 num3 = new Calculator2();

    num3 = num1 + num2;

    num1.display(); // виведе 200
    num2.display(); // виведе 40
    num3.display(); // виведе 240
}
```

■ Переваги від перевантаження операторів:

- Доповнюється функціональність операторів C# при роботі з користувацькими типами даних.
- Оператори можуть розглядатись як функції всередині компілятора.

Перевантаження операторів

ОПЕРАТОРИ	ОПИС
+, -, !, ~, ++, --	Унарні оператори приймають 1 операнд та можуть перевантажуватись
+, -, *, /, %	Бінарні оператори приймають два операнди та можуть перевантажуватись
==, !=, =	Оператори порівняння можуть перевантажуватись
&&,	Умовні логічні оператори не можуть перевантажуватись напрямку
+=, -=, *=, /=, %+=, =	Оператори присвоєння не можуть перевантажуватись



ДЯКУЮ ЗА УВАГУ!

Наступне питання: Перетворення типів та поліморфізм підтипів