Контрольні запитання

**1. Делегат, анонімний делегат, лямбда-вираз**

* **Делегат** — це тип, що представляє посилання на метод з певним сигнатурою. Його можна викликати як функцію.

delegate void Message(); // оголошення

Message m = ShowMessage; // використання

* **Анонімний делегат** — метод без імені, який можна створити прямо під час призначення:

Action greet = delegate() { Console.WriteLine("Привіт!"); };

* **Лямбда-вираз** — скорочена форма анонімного методу:

Action greet = () => Console.WriteLine("Привіт!");

Func<int, int> square = x => x \* x;

**2. Призначення делегатів і поліморфізм**

Делегати дають змогу:

* викликати методи без знання їх імен;
* динамічно змінювати поведінку програми;
* передавати методи як параметри (вищий порядок функцій);
* будувати колекції обробників подій.

**Поліморфізм** означає, що один інтерфейс (делегат) може представляти багато реалізацій (різні методи). Це дозволяє змінювати поведінку без зміни основного коду.

### ****3. Стандартні делегати та приклади****

* **Action** — для методів, які нічого не повертають:

Action sayHello = () => Console.WriteLine("Hello");

* **Func<T1,...,TResult>** — для методів, які повертають значення:

Func<int, int, int> add = (a, b) => a + b;

* **Predicate<T>** — повертає bool, зазвичай для фільтрації:

Predicate<int> isEven = x => x % **2** == **0**;

### ****4. Синтаксис лямбда-виразів і приклади****

Синтаксис:

(параметри) => вираз

**Приклади:**

* Обчислення площі:

Func<double, double, double> rectArea = (w, h) => w \* h;

* Фільтрація масиву:

var even = numbers.Where(x => x % **2** == **0**);

* Підрахунок:

int count = numbers.Count(x => x > **0**);

**5. Зв'язок між подіями й делегатами**

* Подія — це механізм для повідомлення підписаних методів (слухачів), що щось відбулося.
* Делегат — тип, який описує сигнатуру методів-слухачів.
* **Подія використовує делегат, щоб зберігати список обробників.**

**Приклад:**

public **class** **Alarm**

{

public event Action OnAlarm;

public void Ring()

{

Console.WriteLine("🚨 Дзвінок!");

OnAlarm?.Invoke();

}

}

**class** **Program**

{

static void Main()

{

Alarm a = new Alarm();

a.OnAlarm += () => Console.WriteLine("✔️ Спрацював слухач!");

a.Ring();

}

}