Шпаргалка по теме:

C#. Инкапсуляция. Наследование. Полиморфизм.

1. Инкапсуляция

Инкапсуляция — это принцип, при котором данные и методы, которые работают с этими данными, объединяются в одном классе. Доступ к данным контролируется через свойства и методы, что позволяет скрыть детали реализации.

Пример:

```
public class BankAccount
{
    // Приватное поле (инкапсуляция данных)
    private decimal balance;

    // Свойство для контролируемого доступа к полю
    public decimal Balance
    {
        get { return balance; }
        private set // Запрещаем изменение баланса извне
        {
            if (value >= 0)
                balance = value;
        }
    }

    // Метод для пополнения счета
    public void Deposit(decimal amount)
    {
        if (amount > 0)
            Balance += amount;
    }

    // Метод для снятия денег
    public void Withdraw(decimal amount)
    {
        if (amount > 0 && amount <= Balance)
            Balance -= amount;
    }
}
```

Использование:

```
BankAccount account = new BankAccount();
account.Deposit(1000);
account.Withdraw(500);
Console.WriteLine($"Баланс: {account.Balance}");
```

2. Наследование

Наследование — это принцип, при котором один класс (производный) может наследовать поля, свойства и методы другого класса (базового). Это позволяет повторно использовать код и создавать иерархии классов.

Пример:

```
// Базовый класс
public class Vehicle
{
    public string Brand { get; set; }
    public int Speed { get; set; }

    public void Accelerate(int increment)
    {
        Speed += increment;
    }
}
// Производный класс
public class Car : Vehicle
{
    public int NumberOfDoors { get; set; }

    public void DisplayInfo()
    {
        Console.WriteLine($"Марка: {Brand}, Скорость: {Speed}, Двери: {NumberOfDoors}");
    }
}
```

Использование:

```
Car myCar = new Car();
myCar.Brand = "Toyota";
myCar.Speed = 60;
myCar.NumberOfDoors = 4;
myCar.Accelerate(20); // Увеличиваем скорость
myCar.DisplayInfo();
```

3. Полиморфизм

Полиморфизм — это принцип, при котором методы могут работать с объектами разных классов, если эти классы имеют общий базовый класс или интерфейс. Это позволяет вызывать методы, не зная точного типа объекта.

Пример:

```
// Базовый класс
public class Animal
{
    public virtual void MakeSound()
    {
        Console.WriteLine("Звук животного");
    }
}

// Производные классы
public class Dog : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("Гав-гав!");
    }
}

public class Cat : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("Мяу!");
    }
}
```

Использование:

```
Animal myAnimal = new Animal();
Animal myDog = new Dog();
Animal myCat = new Cat();

myAnimal.MakeSound(); // Звук животного
myDog.MakeSound();    // Гав-гав!
myCat.MakeSound();    // Мяу!
```

Пример использования в Windows Forms

Инкапсуляция

```
public partial class MainForm : Form
{
    private int counter = 0; // Приватное поле (инкапсуляция)

    public MainForm()
    {
        InitializeComponent();
    }

    // Метод для обновления счетчика
    private void UpdateCounter()
    {
        counter++;
        label1.Text = $"Количество нажатий: {counter}";
    }

    // Обработчик нажатия кнопки
    private void button1_Click(object sender, EventArgs e)
    {
        UpdateCounter();
    }
}
```

Наследование

```
// Базовый класс для формы
public class BaseForm : Form
{
    protected Button button;
    protected Label label;

    public BaseForm()
    {
        button = new Button { Text = "Нажми меня", Location = new Point(10, 10) };
        label = new Label { Location = new Point(10, 50), AutoSize = true };

        button.Click += Button_Click;
        Controls.Add(button);
        Controls.Add(label);
    }
    protected virtual void Button_Click(object sender, EventArgs e)
    {
        label.Text = "Нажата кнопка в базовой форме";
    }
}

// Производный класс
public class DerivedForm : BaseForm
{
    public DerivedForm()
    {
        button.Text = "Новая кнопка";
    }

    protected override void Button_Click(object sender, EventArgs e)
    {
        label.Text = "Нажата кнопка в производной форме";
    }
}
```

Полиморфизм

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Animal myAnimal = new Dog(); // Полиморфизм
        myAnimal.MakeSound(); // Вызов метода Dog.MakeSound()
    }
}
```