**Принцип инверсии зависимостей** (Dependency Inversion Principle) служит для создания слабосвязанных сущностей, которые легко тестировать, модифицировать и обновлять. Этот принцип можно сформулировать следующим образом:

**Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те и другие должны зависеть от абстракций.**

**Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.**

Чтобы понять принцип, рассмотрим следующий пример:

class Book

{

    public string Text { get; set; }

    public ConsolePrinter Printer { get; set; }

    public void Print()

    {

        Printer.Print(Text);

    }

}

class ConsolePrinter

{

    public void Print(string text)

    {

        Console.WriteLine(text);

    }

}

Класс Book, представляющий книгу, использует для печати класс ConsolePrinter. При подобном определении класс Book зависит от класса ConsolePrinter. Более того мы жестко определили, что печать книгу можно только на консоли с помощью класса ConsolePrinter. Другие же варианты, например, вывод на принтер, вывод в файл или с использованием каких-то элементов графического интерфейса - все это в данном случае исключено. Абстракция печати книги не отделена от деталей класса ConsolePrinter. Все это является нарушением принципа инверсии зависимостей.

Теперь попробуем привести наши классы в соответствие с принципом инверсии зависимостей, отделив абстракции от низкоуровневой реализации:

interface IPrinter

{

    void Print(string text);

}

class Book

{

    public string Text { get; set; }

    public IPrinter Printer { get; set; }

    public Book(IPrinter printer)

    {

        this.Printer = printer;

    }

    public void Print()

    {

        Printer.Print(Text);

    }

}

class ConsolePrinter : IPrinter

{

    public void Print(string text)

    {

        Console.WriteLine("Печать на консоли");

    }

}

class HtmlPrinter : IPrinter

{

    public void Print(string text)

    {

        Console.WriteLine("Печать в html");

    }

}

Теперь абстракция печати книги отделена от конкретных реализаций. В итоге и класс Book и класс ConsolePrinter зависят от абстракции IPrinter. Кроме того, теперь мы также можем создать дополнительные низкоуровневые реализации абстракции IPrinter и динамически применять их в программе:

Book book = new Book(new ConsolePrinter());

book.Print();

book.Printer = new HtmlPrinter();

book.Print();