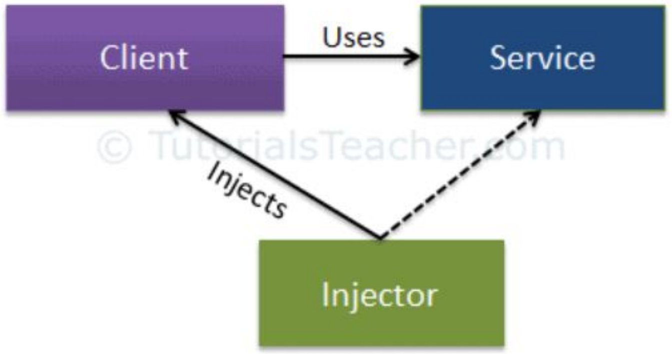
Dependency Injection

Group 3

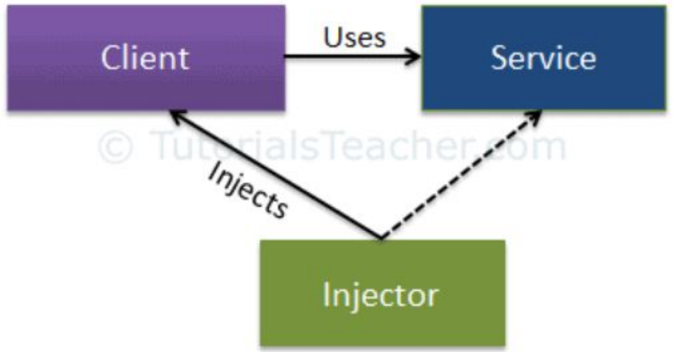
Definition

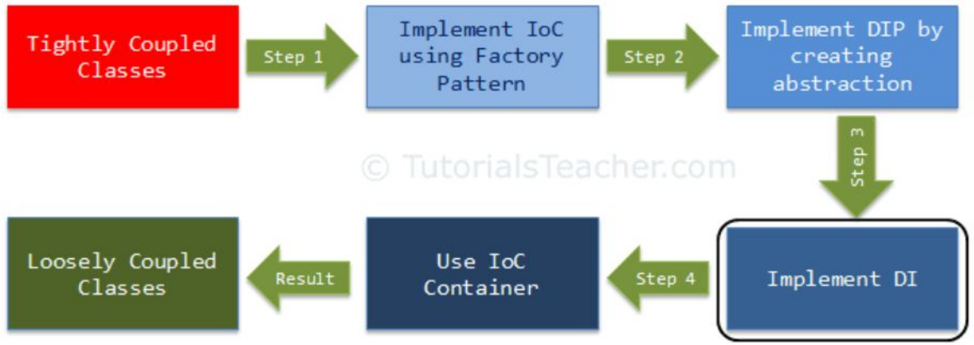
- Client class
    - dependent class that depends on Service class
- Service class
    - dependency that provides service to the Client class



Dependency Injection

- Used in Inversion Of Control (IOC)
- Decouple Service from Client
- Injecter injects Service into Client
- Client dont change when requirement change
- Injector can inject multiple kinds of Service

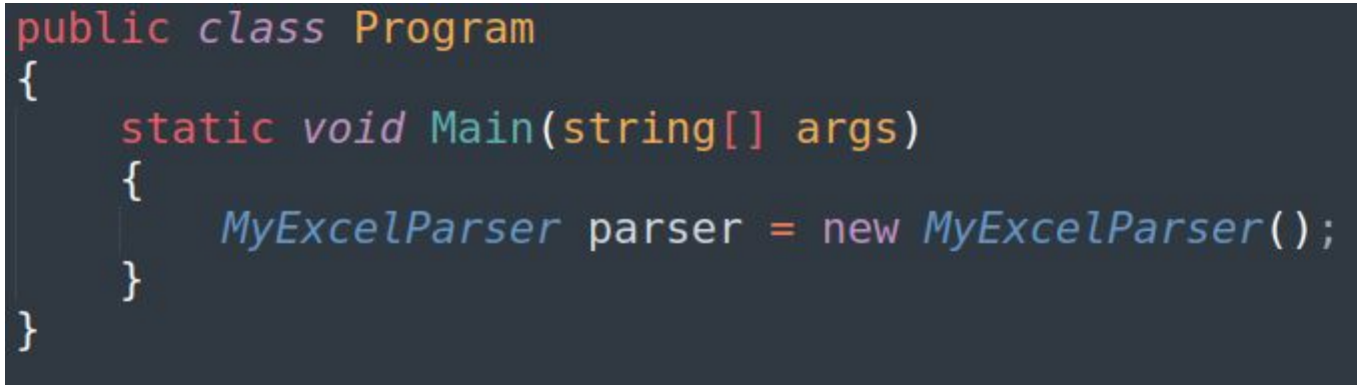




Type of Dependency injection

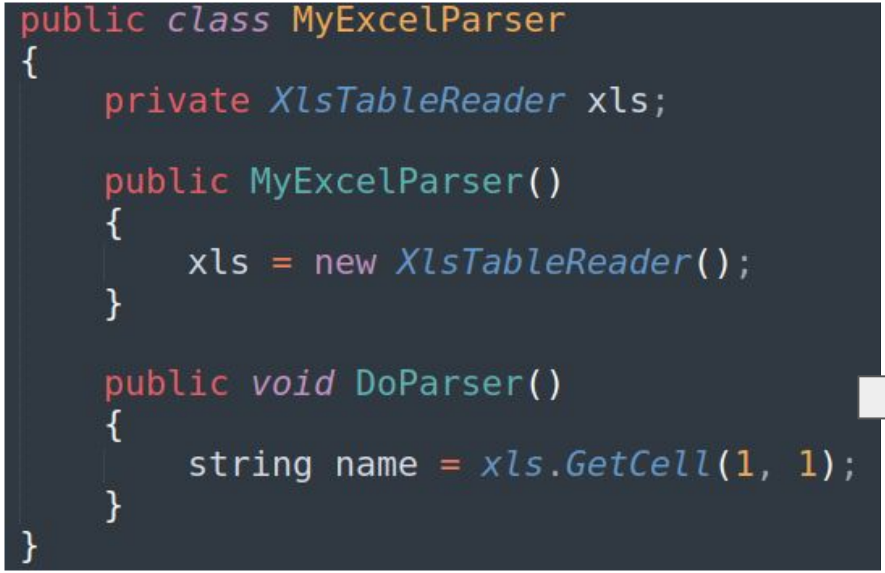
- Contructor
  - new Client(new ServiceA())
- Property
  - client.service = new ServiceA()
- Method
  - client.setService(new ServiceA)

Example: Write a Excel Parser

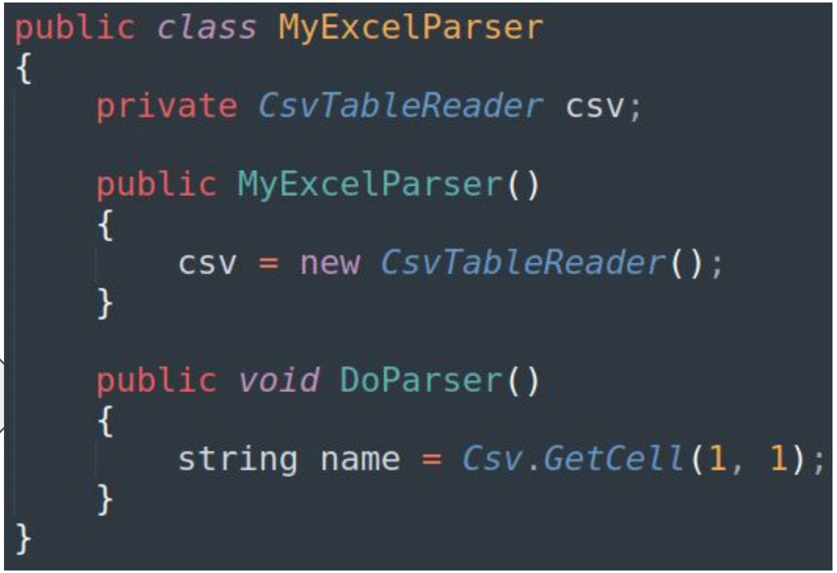


```
public class Program
{
    static void Main(string[] args)
    {
        MyExcelParser parser = new MyExcelParser();
    }
}
```

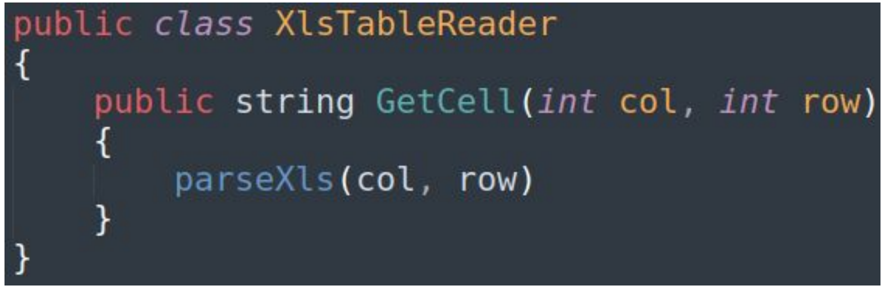
Excel格式: .xls .csv .xlsx .xlsm…



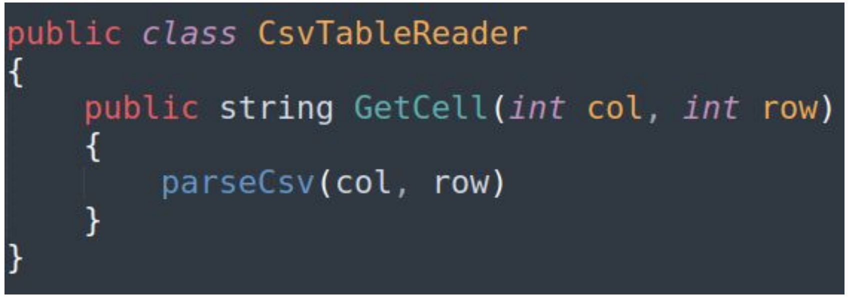
```
public class MyExcelParser
{
    private XlsTableReader xls;

    public MyExcelParser()
    {
        xls = new XlsTableReader();
    }

    public void DoParser()
    {
        string name = xls.GetCell(1, 1);
    }
}
```



```
public class MyExcelParser
{
    private CsvTableReader csv;

    public MyExcelParser()
    {
        csv = new CsvTableReader();
    }

    public void DoParser()
    {
        string name = Csv.GetCell(1, 1);
    }
}
```

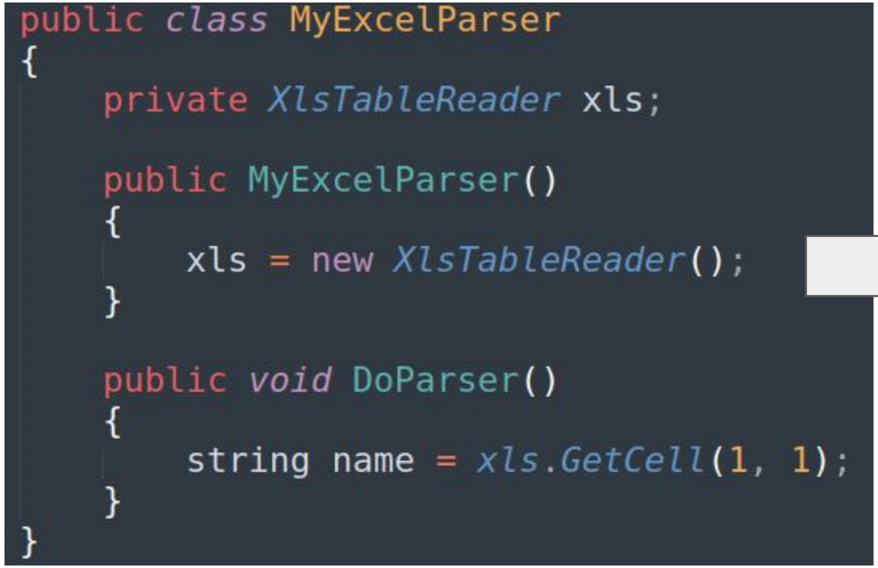


```
public class XlsTableReader
{
    public string GetCell(int col, int row)
    {
        parseXls(col, row)
    }
}
```

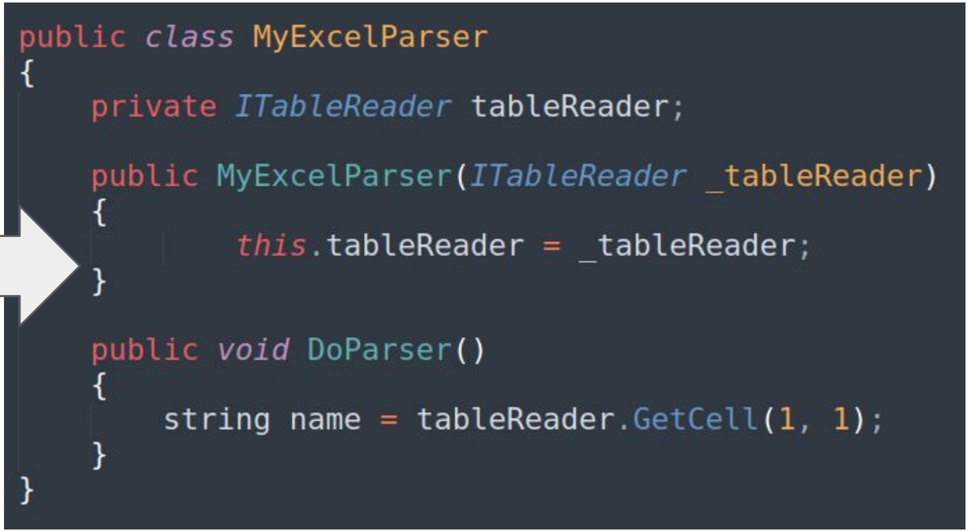


```
public class CsvTableReader
{
    public string GetCell(int col, int row)
    {
        parseCsv(col, row)
    }
}
```

What if we…

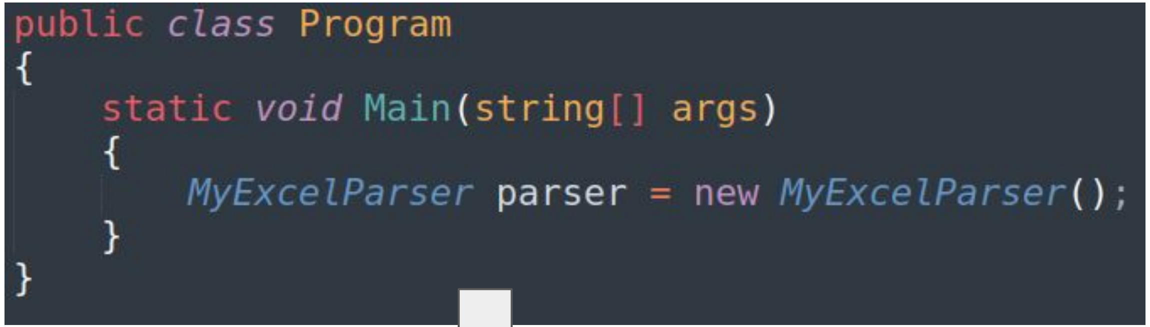


```
public class MyExcelParser
{
    private XlsTableReader xls;

    public MyExcelParser()
    {
        xls = new XlsTableReader();
    }

    public void DoParser()
    {
        string name = xls.GetCell(1, 1);
    }
}
```

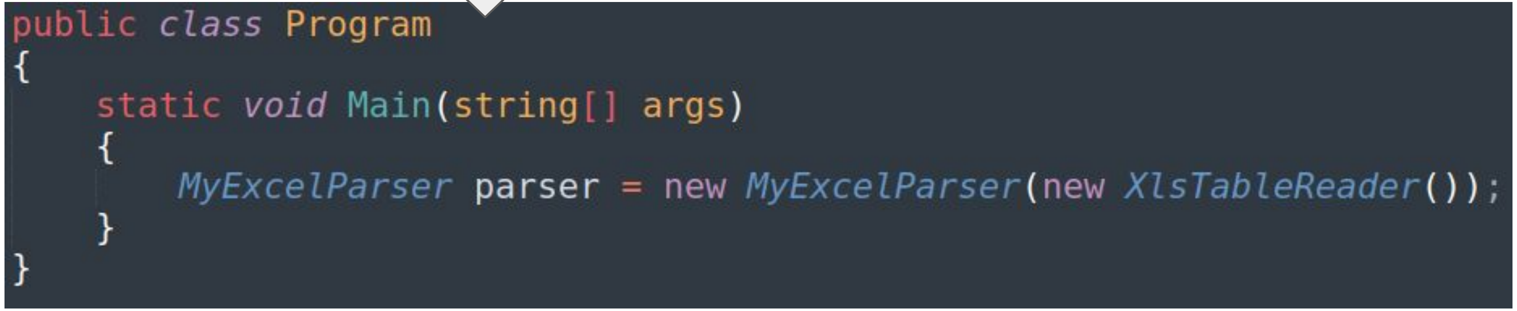


```
public class MyExcelParser
{
    private ITableReader tableReader;

    public MyExcelParser(ITableReader _tableReader)
    {
            this.tableReader = _tableReader;
    }

    public void DoParser()
    {
        string name = tableReader.GetCell(1, 1);
    }
}
```

And

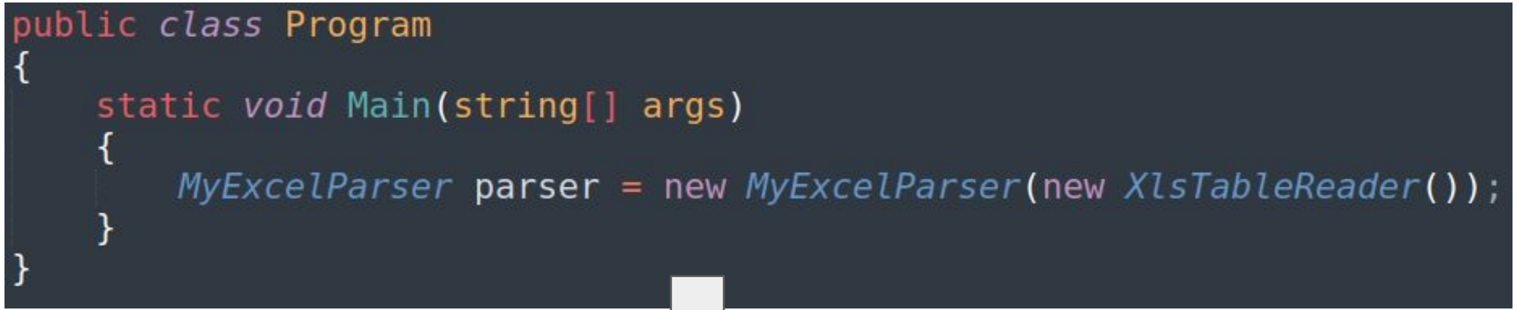


```
public class Program
{
    static void Main(string[] args)
    {
        MyExcelParser parser = new MyExcelParser();
    }
}
```

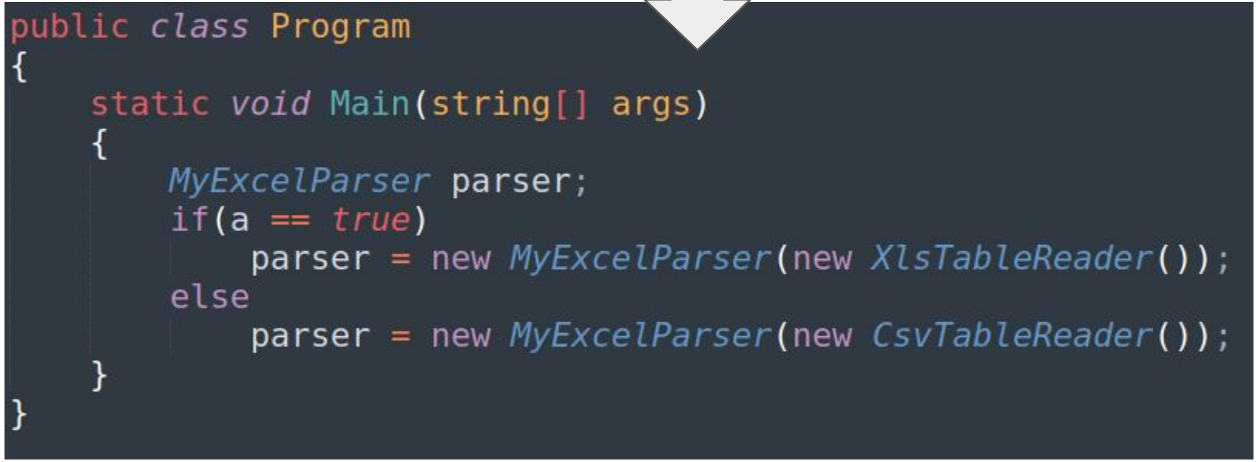


```
public class Program
{
    static void Main(string[] args)
    {
        MyExcelParser parser = new MyExcelParser(new XlsTableReader());
    }
}
```

Then we can



```
public class Program
{
    static void Main(string[] args)
    {
        MyExcelParser parser = new MyExcelParser(new XlsTableReader());
    }
}
```



```
public class Program
{
    static void Main(string[] args)
    {
        MyExcelParser parser;
        if(a == true)
            parser = new MyExcelParser(new XlsTableReader());
        else
            parser = new MyExcelParser(new CsvTableReader());
    }
}
```

Client? Service?

- Client:
  - MyExcelParser()
- Service:
  - XlsTableReader()
  - CsvTableReader()
- Injector
  - Main()
- Inject Service into Client:
  - MyExcelParser(new XlsTableReader())
  - MyExcelParser(new CsvTableReader())

Benefits

- Write less code
- Reduce coupling
- Easier to write new modules: .xlsx, .txt….

Thank you