Abstract Factory Pattern

관련 부품을 조합해서 인스턴스 생성

- 부품의 구체적인 구현에는 주목하지 않고, **인터페이스(API)에 주목**.
- **추상적인 공장**(Abstract Factory)에서 **추상적인 부품**을 조립하고 **추상적인 제품**을 완성한다.
    - 많은 수의 연관된 하위 클래스를 특정 그룹으로 묶어 한번에 교체.
      ex) 특정 라이브러리 배포시 OS별로 지원하는 기능이 다를 때, OS별 기능 변경을 통합적으로 변경.

Architecture, Modeling, DesignPattern, Creational

Factory

Abstract Factory

```
package factory;

public abstract class Factory {
    public static Factory getFactory(String className) {
        Factory factory = null;
        try {
            factory = (Factory) Class.forName(className).newInstance();
        } catch (InstantiationException | IllegalAccessException |
ClassNotFoundException e) {
            e.printStackTrace();
        }
        return factory;
    }
    public abstract LinkItem createLink(String caption, String url);
    public abstract TrayItem createTray(String caption);
    public abstract Page createPage(String caption, String author);
}
```

Abstract Products

```
public abstract class Item {
    protected String caption;

    public Item(String caption) {
        super();
        this.caption = caption;
    }
    public abstract String makeHTML();
```

```
}
```

```
package factory;

public abstract class LinkItem extends Item {

    protected String url;
    public LinkItem(String caption, String url) {
        super(caption);
        this.url = url;
    }
}
```

```
package factory;

import java.util.ArrayList;

public abstract class TrayItem extends Item {

    protected final ArrayList<Item> tray = new ArrayList<>();
    public TrayItem(String caption) {
        super(caption);
    }
    public void add(Item item) {
        tray.add(item);
    }
}
```

```
package factory;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;

public abstract class Page {
    protected String title;
    protected String author;
    protected ArrayList<Item> content = new ArrayList<>();
    public Page(String title, String author) {
        this.title = title;
```

```
        this.author = author;
    }
    public void add(Item item) {
        content.add(item);
    }
    public void output() {
        System.out.println(this.makeHTML());
        try (BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(new FileOutputStream(title + ".html"),
StandardCharsets.UTF_8))) {
            bw.write(this.makeHTML());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public abstract String makeHTML();
}
```

Concrete Factories

Concrete Factory1 - ListFactory

Concrete Factory

```
package listfactory;

import factory.Factory;
import factory.LinkItem;
import factory.Page;
import factory.TrayItem;

public class ListFactory extends Factory {

    @Override
    public LinkItem createLink(String caption, String url) {
        return new ListLink(caption, url);
    }

    @Override
    public TrayItem createTray(String caption) {
        return new ListTray(caption);
    }

    @Override
    public Page createPage(String caption, String author) {
        return new ListPage(caption, author);
```

```
    }

}
```

Concrete Products

```
package listfactory;

import factory.LinkItem;

public class ListLink extends LinkItem {
    public ListLink(String caption, String url) {
        super(caption, url);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
        sb.append("<li>");
        sb.append("<a href=\"");
        sb.append(url);
        sb.append("\">");
        sb.append(caption);
        sb.append("</a>");
        sb.append("</li>\n");
        return sb.toString();
    }

}
```

```
package listfactory;

import factory.TrayItem;

public class ListTray extends TrayItem {

    public ListTray(String caption) {
        super(caption);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
```

```
        sb.append("<li>\n");
        sb.append(caption);
        sb.append("\n<ul>\n");
        tray.forEach(item -> sb.append(item.makeHTML()));
        sb.append("</ul>\n");
        sb.append("</li>");
        return sb.toString();
    }
}
```

```
package listfactory;

import factory.Page;

public class ListPage extends Page {

    public ListPage(String title, String author) {
        super(title, author);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
        sb.append("<html>\n");
        sb.append("<head><title>");
        sb.append(title);
        sb.append("</title></head>\n");
        sb.append("<body>\n");

        sb.append("<h1>");
        sb.append(title);
        sb.append("</h1>");
        sb.append("<ul>\n");
        content.forEach(item -> sb.append(item.makeHTML()));
        sb.append("</ul>\n");
        sb.append("<hr>");
        sb.append("<footer>");
        sb.append(author);
        sb.append("</footer>");
        sb.append("</body>\n</html>");
        return sb.toString();
    }

}
```

Concrete Factory2 - DivFactory

Concrete Factory

```
package divfactory;

import factory.Factory;
import factory.LinkItem;
import factory.Page;
import factory.TrayItem;

public class DivFactory extends Factory {

    @Override
    public LinkItem createLink(String caption, String url) {
        return new DivLink(caption, url);
    }

    @Override
    public TrayItem createTray(String caption) {
        return new DivTray(caption);
    }

    @Override
    public Page createPage(String caption, String author) {
        return new DivPage(caption, author);
    }

}
```

Concrete Products

```
package divfactory;

import factory.LinkItem;

public class DivLink extends LinkItem {

    public DivLink(String caption, String url) {
        super(caption, url);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
```

```
        sb.append("\n<span>");
        sb.append("<a href=\"");
        sb.append(url);
        sb.append("\">");
        sb.append(caption);
        sb.append("</a>");
        sb.append("</span>\n");
        return sb.toString();
    }
}
```

```
package divfactory;

import factory.TrayItem;

public class DivTray extends TrayItem {

    public DivTray(String caption) {
        super(caption);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
        sb.append("\n<div style=\"border: 1px solid black; ; margin: 5px;
padding: 5px\">");
        sb.append("\n<div>");
        sb.append(caption);
        sb.append("</div>\n");
        sb.append("\n<div>");
        tray.forEach(item -> sb.append(item.makeHTML()));
        sb.append("</div>\n");
        sb.append("</div>\n");
        sb.append("<br>\n");
        return sb.toString();
    }

}
```

```
package divfactory;

import factory.Page;

public class DivPage extends Page {

    public DivPage(String title, String author) {
```

```
        super(title, author);
    }

    @Override
    public String makeHTML() {
        StringBuilder sb = new StringBuilder();
        sb.append("<html>\n");
        sb.append("<head><title>");
        sb.append(title);
        sb.append("</title></head>\n");
        sb.append("<body>\n");

        sb.append("<h1>");
        sb.append(title);
        sb.append("</h1>");
        sb.append("\n<div>");
        content.forEach(item -> sb.append(item.makeHTML()));
        sb.append("</div>\n");
        sb.append("<hr>");
        sb.append("<footer>");
        sb.append(author);
        sb.append("</footer>");
        sb.append("</body>\n</html>");
        return sb.toString();
    }

}
```

Client

- Client는 구체적인 부품, 제품, 공장에 대해 모른다!
  - API만 이용! 말그대로 추상적인 공장(Abstract Factory)에서 추상적인 부품을 조립하고 추상적인 제품을 완성한다.

```
import factory.Factory;
import factory.LinkItem;
import factory.Page;
import factory.TrayItem;

public class Main {
    public static void main(String[] args) {
        operateFactory("listfactory.ListFactory");
        operateFactory("divfactory.DivFactory");
    }
    private static void operateFactory(String className) {
        Factory factory = Factory.getFactory(className);
```

```
        LinkItem link_todayHumor = factory.createLink("오늘의 유머",
"http://www.todayhumor.co.kr");
        LinkItem link_ruliweb = factory.createLink("루리웹",
"http://ruliweb.com/");
        TrayItem trary_community = factory.createTray("커뮤니티");
        trary_community.add(link_todayHumor);
        trary_community.add(link_ruliweb);
        LinkItem link_google = factory.createLink("Google",
"http://www.google.com");
        LinkItem link_daum = factory.createLink("Daum",
"http://www.daum.net");
        LinkItem link_naver = factory.createLink("Naver",
"http://www.naver.com");
        LinkItem link_naver_m = factory.createLink("Naver Mobile",
"http://m.naver.com");

        TrayItem tray_naver = factory.createTray("Naver");
        tray_naver.add(link_naver);
        tray_naver.add(link_naver_m);
        TrayItem tray_searchEngine = factory.createTray("검색 엔진");
        tray_searchEngine.add(link_google);
        tray_searchEngine.add(link_daum);
        tray_searchEngine.add(tray_naver);
        Page page = factory.createPage(className, "By ledyx");
        page.add(trary_community);
        page.add(tray_searchEngine);
        page.output();
    }
}
```

실행 결과

Concrete Factory1 - ListFactory

```
<html>
<head><title>listfactory.ListFactory</title></head>
<body>
<h1>listfactory.ListFactory</h1><ul>
<li>
커뮤니티
<ul>
<li><a href="http://www.todayhumor.co.kr">오늘의 유머</a></li>
<li><a href="http://ruliweb.com/">루리웹</a></li>
</ul>
</li><li>
검색 엔진
<ul>
```

```
<li><a href="http://www.google.com">Google</a></li>
<li><a href="http://www.daum.net">Daum</a></li>
<li>
Naver
<ul>
<li><a href="http://www.naver.com">Naver</a></li>
<li><a href="http://m.naver.com">Naver Mobile</a></li>
</ul>
</li></ul>
</li></ul>
<hr><footer>By ledyx</footer></body>
</html>
```

<html> <head><title>listfactory.ListFactory</title></head> <body> <h1>listfactory.ListFactory</h1> 커뮤니티 <ul> 오늘의 유머 루리웹   검색 엔진 <ul> Google Daum Naver <ul> Naver Naver Mobile   <hr><footer>By ledyx</footer></body> </html>

Concrete Factory2 - DivFactory

```
<html>
<head><title>divfactory.DivFactory</title></head>
<body>
<h1>divfactory.DivFactory</h1>
<div>
<div style="border: 1px solid black; ; margin: 5px; padding: 5px">
<div>커뮤니티</div>

<div>
<span><a href="http://www.todayhumor.co.kr">오늘의 유머</a></span>

<span><a href="http://ruliweb.com/">루리웹</a></span>
</div>
</div>
<br>

<div style="border: 1px solid black; ; margin: 5px; padding: 5px">
<div>검색 엔진</div>

<div>
<span><a href="http://www.google.com">Google</a></span>
```

```
<span><a href="http://www.daum.net">Daum</a></span>

<div style="border: 1px solid black; ; margin: 5px; padding: 5px">
<div>Naver</div>

<div>
<span><a href="http://www.naver.com">Naver</a></span>

<span><a href="http://m.naver.com">Naver Mobile</a></span>
</div>
</div>
<br>
</div>
</div>
<br>
</div>
<hr><footer>By ledyx</footer></body>
</html>
```

From:
https://ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://ledyx.xyz/doku.php?id=design_pattern:abstract_factory_pattern&rev=1595817177

Last update: **2026/10/24 07:03**