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.

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;

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;

public abstract class LinkItem extends Item {

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

```
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();
}
```

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

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=design_pattern:abstract_factory_pattern&rev=1509203237

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