Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4
з дисципліни **«Технології розроблення програмного забезпечення»**
на тему «шаблони проектування «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY» »
Тема для лабораторного циклу «HTTP-сервер»

Виконала

студентка групи ІА-04
Глушко
Юлія Петрівна

Перевірив

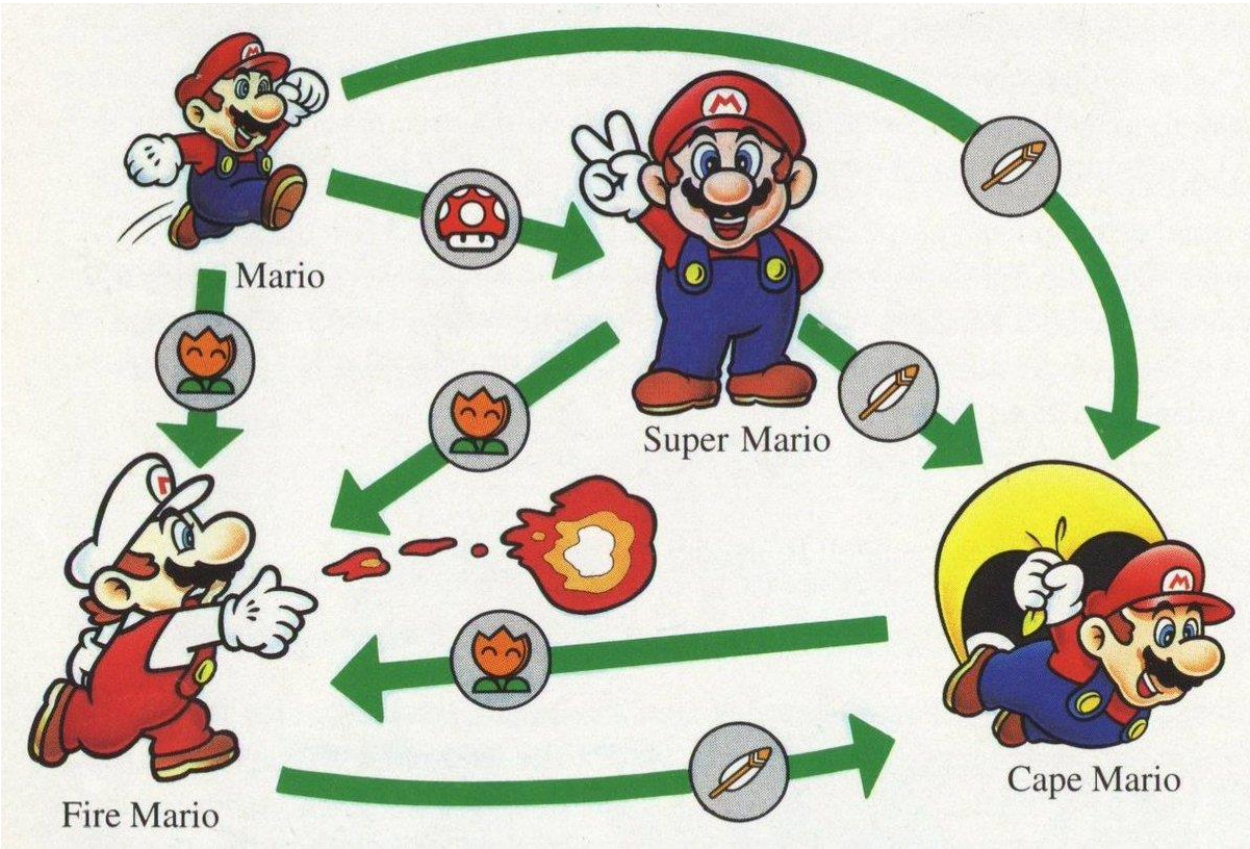
Колеснік Валерій
Миколайович

Защищено з балом _____

Київ 2022

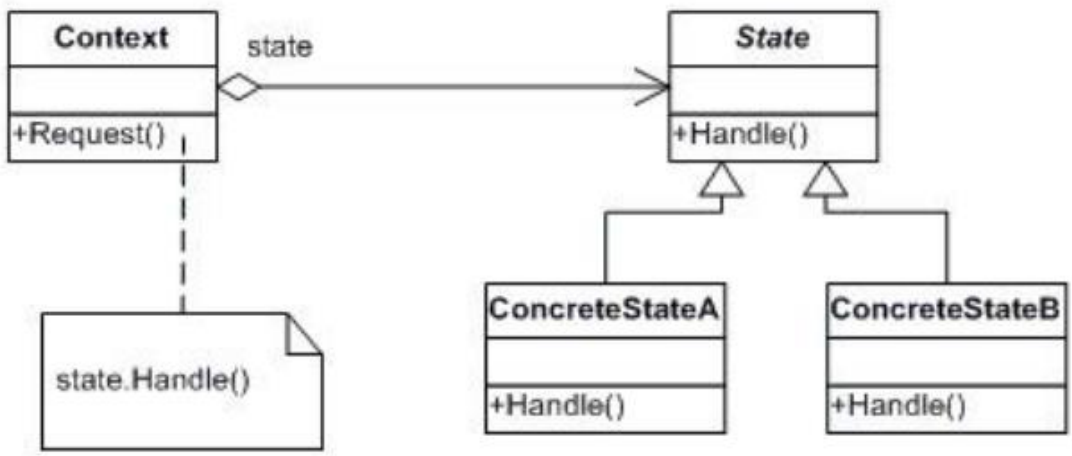
Хід роботи :

*Шаблон «**State**»*



Теоретичні відомості

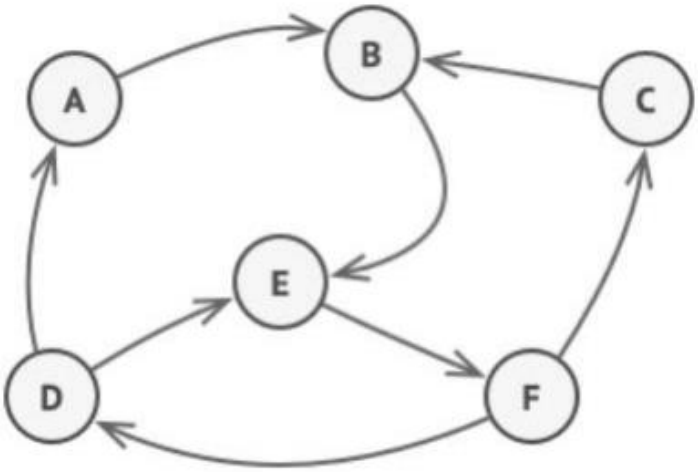
- Структура :



- Призначення патерну:

  Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства - бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс. Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта. Це дозволяє легко додавати в майбутньому і обробляти нові стани, відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

  Патерн Стан неможливо розглядати у відриві від концепції машини станів, також відомої як стейт-машина або кінцевий автомат.



Основна ідея полягає в тому, що програма може знаходитися в одному з декількох станів, які весь час змінюють один одного. Набір цих станів, а також переходів між ними, зумовлений і кінцевий. Перебуваючи в різних станах, програма може по-різному реагувати на одні і ті ж події, які відбуваються з нею. Плутанина і нагромадження умов особливо сильно проявляється в старих проектах. Набір можливих станів буває важко визначити заздалегідь, тому вони весь час додаються в процесі еволюції програми. Через це рішення, яке виглядало простим і ефективним на самому початку розробки, може згодом стати проекцією великого макаронного монстра

Патерн Стан пропонує створити окремі класи для кожного стану, в якому може перебувати об'єкт, а потім винести туди поведінки, які відповідають цим станам. Замість того, щоб зберігати код всіх станів, початковий об'єкт, званий контекстом, буде містити посилання на один з об'єктів-станів і делегувати йому роботу, залежну від стану. Завдяки тому, що об'єкти станів матимуть загальний інтерфейс, контекст зможе делегувати роботу станом, не прив'язуючись до його класу. Поведінка контексту можна буде змінити в будь-який момент, підключивши до нього інший об'єкт-стан.

- Переваги та недоліки:
  + Позбавляє від безлічі великих умовних операторів машини станів.
  + Концентрує в одному місці код, пов'язаний з певним станом.
  + Спрощує код контексту.
  - Може невиправдано ускладнити код, якщо станів мало і вони рідко змінюються.

Застосування при реалізації лабораторного циклу

При виконанні лабораторно циклу патерн «State» був застосований при реалізації динамічних web-сторінок .

Життєвий цикл jsp сторінки включає досить багато станів(рис. 1), які переходять з одного в інший. Для спрощення коду було створено окремі класи для кожного стану, в яких може перебувати об'єкт, та винесено всю логіку, пов’язану з певним станом в самі класи станів. Завдяки цьому код стає значно краще виглядати , виконується принцип єдиного обов'язку(на кожен об'єкт має бути покладений один єдиний обов'язок.), код не нагромаджується в одному місці.

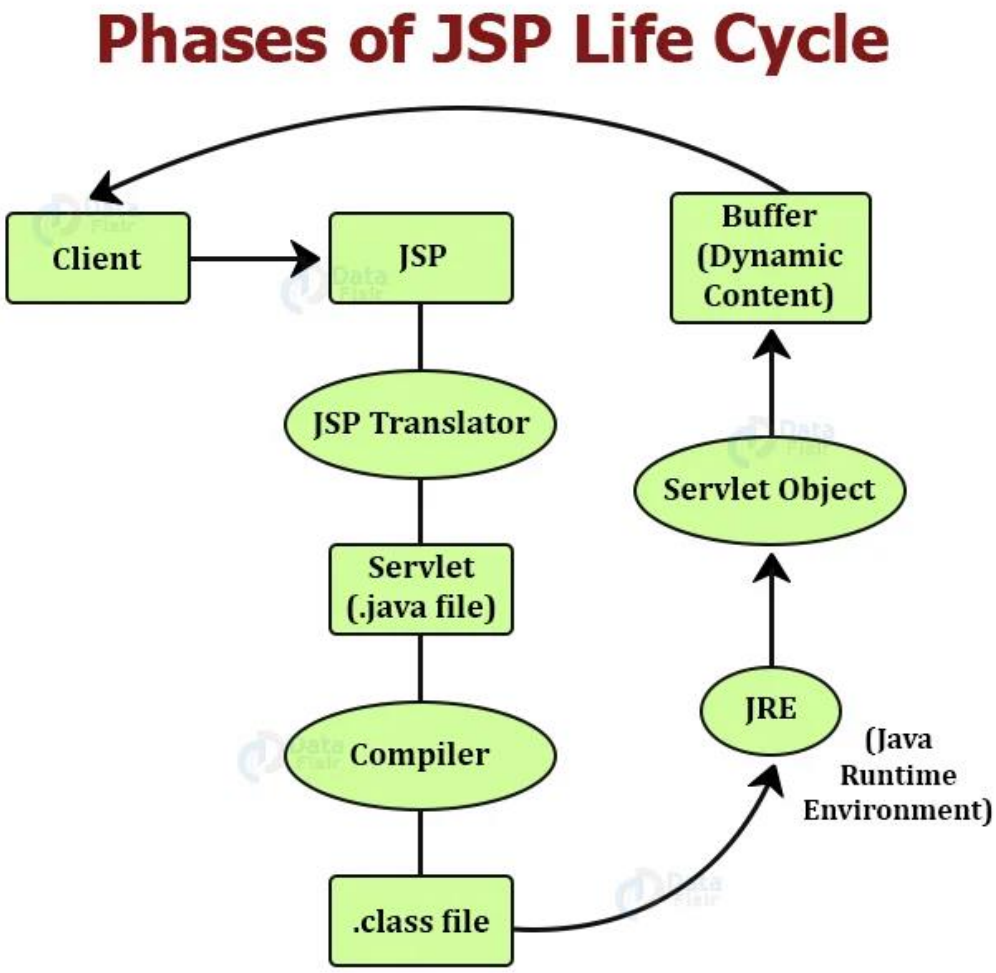


Рис. 1- Життєвий цикл jsp сторінки

Схема взаємодії класів при реалізації лабораторного циклу зображена на рисунку 2

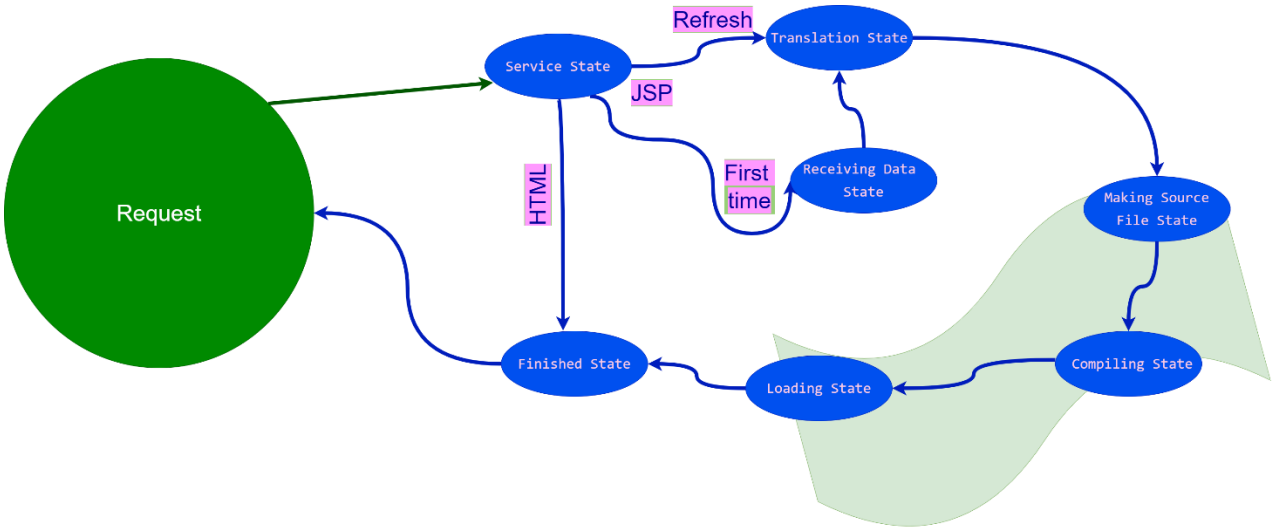


Рис. 2- Схема взаємодії класів

Діаграма класів, оптимізованих за допомогою патерну «State», зображена на рис.3

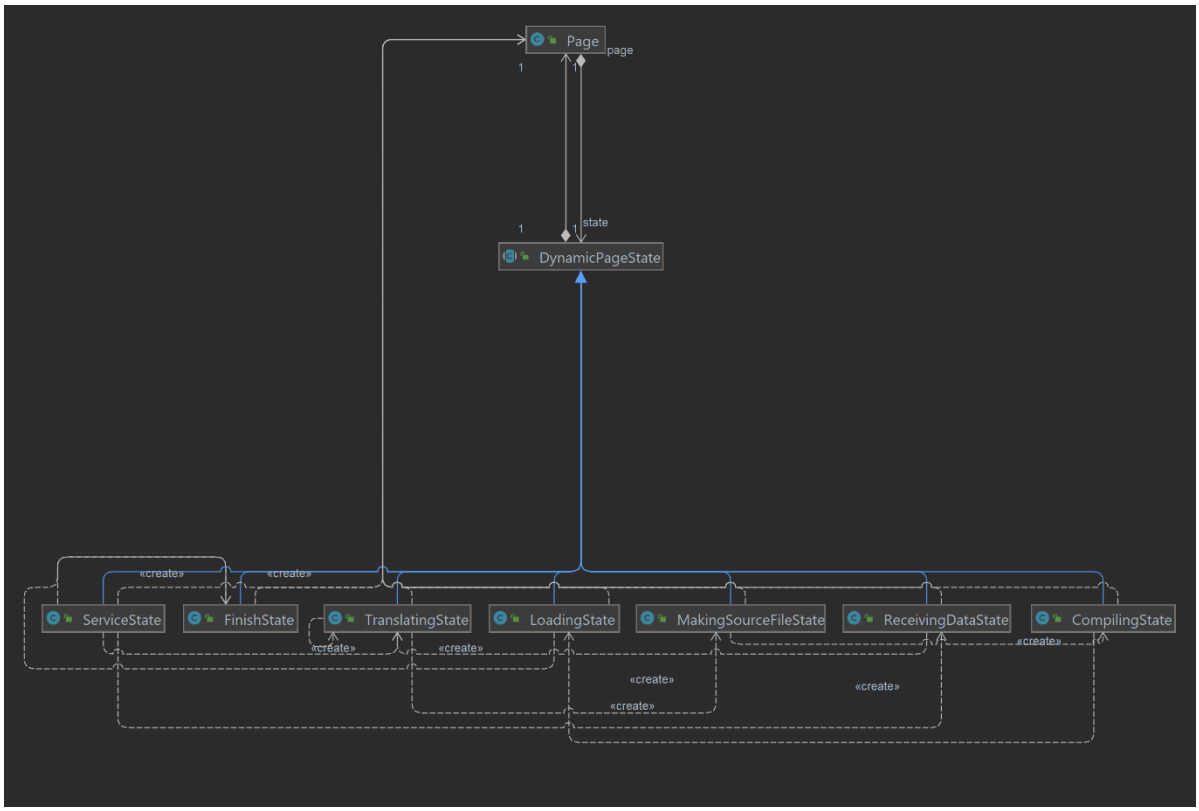


Рис.3 – діаграма класів, що реалізують шаблон «State»

У нашому випадку взаємодія користувача із сервером відбувається через запити/відповіді. Динамічна генерація веб-сторінок застосовується, якщо користувач зробив запит на .jsp сторінку. У випадку, якщо був зроблений запит на статичну сторінку (.html), стани які відповідають за обробку динамічних сторінок, пропускаються.

Процес читання шаблону з файлу певної сторінки займає деякий час, тому для економії ресурсів кожна запитана сторінка має поле для прочитаних та ще не оброблених даних динамічної сторінки. Це зекономить час і ресурси, якщо користувач буде багаторазово оновлювати сторінку. Тому якщо користувач оновлює сторінку, стан читання даних з файлу з шаблоном сторінки пропускається.

Щоб не перевантажувати код класу Request, було створено ServiceState, в якому відбувається визначення, який стан буде наступним, в залежності від того, динамічна чи статична сторінка і якщо динамічна, то чи це перше відвідування сторінки, чи її оновлення .

```
 //якщо сторінка динамічна
if (page.getPathToFile().split("\\.")[1].equals("jsp"))
{
    //якщо це перше відвідування сторінки
    if (page.getPageData() == null) {
        super.page.changeState(new
ReceivingDataState(super.page));
    } else {
        //якщо це оновлення сторінки
        super.page.changeState(new
TranslatingState(super.page));
    }
} else {
    //якщо сторінка статична
    super.page.changeState(new
```

```
FinishState(super.page));
}
```

TranslatingState, MakingSourceFileState, LoadingState, CompilingState виконуються послідовно і були виокремлені в окремі стани задля збереження принципу єдиного обов’язку .

Сама сутність Page містить метод для зміни стану

```
public void changeState(DynamicPageState state){
    this.state=state;
    state.doAction();
}
```

Висновок: під час виконання лабораторної роботи було розглянуто патерни проектування «singleton», «iterator», «proxy», «state», «strategy». Патерн «state» був застосований як частина структури лабораторного практикуму