++rr

**МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота № 2**

**з дисципліни “Об’єктно орієнтоване програмування”**

**тема “Реалізація складних структурних шаблонів проектування”**

|  |  |  |
| --- | --- | --- |
| **Виконав**  **студент II курсу**  **групи КП-83**  **Коваль Андрій Олекснадрович**  **(прізвище, ім’я, по батькові)** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **(прізвище, ім’я, по батькові)** |

**Київ 2020**

**Мета роботи**

Ознайомлення з основними характеристиками шаблонів «Заступник», «Фасад», «Адаптер» та «Легковаговик», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

**Постановка завдання**

Варіант 8

1. Розробити модуль до програмного забезпечення роботи бухгалтерії, який буде автоматично формувати пакет електронних фінансових документів (акт виконаних робіт, рахунок-фактура тощо), що підтверджують виконання обов’язків компанії перед замовником, тільки у разі наявності оплати останнім послуги чи товару.

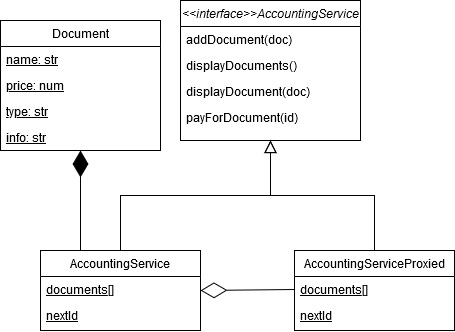
2. Стара система автоматичного вітання користувачів сайту раніше всіх вітала однаково: «Привіт, » + ім’я людини. Тепер вирішено, що таким чином система повинна вітати тільки людей, молодше 35 років. Користувачі, старші 35 років, повинні побачити повідомлення «Доброго дня, » + ПІБ. За допомогою шаблона проектування забезпечити диференціацію виведення вітального рядка (передачу до відповідного метода цілого об’екта Користувач, або значень його полів ПІБ та вік). У разі, якщо вік користувача невідомий, за замовчанням вважати його більшим, ніж 35 років.

**Задача 1**

**Обґрунтування вибору шаблону**

Шаблон «Заступник» використовується для захисту доступу до якогось роду даних та методів. У нашому випадку необхідно не видавати документи, оплата за які не була проведена.

**Діаграма класів**



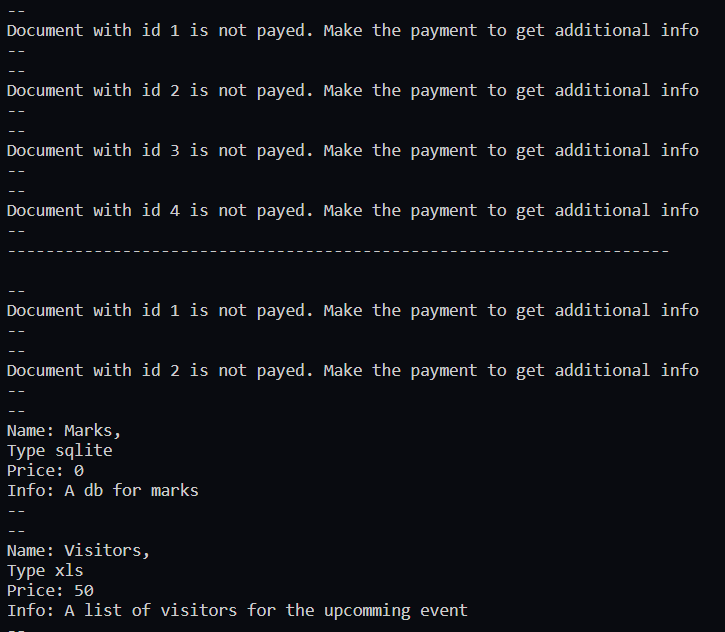
**Фрагменти коду програми**

|  |
| --- |
| **Інтерфейс системи документів** |
| import { Document } from './document';  type accountedDocument = { id: number, item: Document, isPaid: boolean };  export default accountedDocument;  export interface IAccouningService {  addDocument(document: Document | accountedDocument): void;  displayDocuments(): void;  displayDocument(doc: number | accountedDocument): void;  payForDocument(id: number): void;  } |

|  |
| --- |
| **Документ** |
| export class Document {  constructor(public name: string,  public price: number,  public type: string,  public info: string){}  } |
| **Сервіс обліку документів** |
| import { Document } from './document';  import accountedDocument, { IAccouningService } from './AccountingInterface';  export class AccountingService implements IAccouningService {  public static documents: accountedDocument[] = [];  static nextId: number = 0;    addDocument(document: Document | accountedDocument) {  if (document instanceof Document) {  const doc: accountedDocument = { id: ++AccountingService.nextId, item: document as Document, isPaid: false };  AccountingService.documents.push(doc);  } else {  AccountingService.documents.push(document as accountedDocument);  }  }  payForDocument(id: number): void {  for (const doc of AccountingService.documents) {  if (doc.id === id) {  doc.isPaid = true;  return;  }  }  throw new Error('Document with id '+ id + ' not found');  }  displayDocuments() {  console.log('All documents');  AccountingService.documents.forEach(this.displayDocument);  console.log('All documents');  }  displayDocument(doc: number | accountedDocument) {  console.log('--');  const document: accountedDocument = typeof doc === 'number' ? AccountingService.documents.filter(d => d.id !== doc)[0] || null : doc;  if (document == null) {  console.log(`Document not found`);  return;  }  console.log(`Name: ${document.item.name},\nType ${document.item.type}\nPrice: ${document.item.price}\nInfo: ${document.item.info}`);  console.log('--');  }  } |

|  |
| --- |
| **Сервіс із заступником для контролю оплати** |
| import { Document } from './document';  import accountedDocument, { IAccouningService } from './AccountingInterface';  import { AccountingService } from './AccountingService';  export class AccountingServiceProxied implements IAccouningService {  static accountingService = new AccountingService();  static get documents() {  return AccountingService.documents;  }  addDocument(document: Document | accountedDocument): void {  AccountingServiceProxied.accountingService.addDocument(document);  }  displayDocuments(): void {  AccountingService.documents.forEach(this.displayDocument);  }  displayDocument(doc: number | accountedDocument): void {  const document: accountedDocument = typeof doc === 'number' ? AccountingService.documents.filter(d => d.id !== doc)[0] || null : doc;  if (document.isPaid) {  AccountingServiceProxied.accountingService.displayDocument(doc);  } else {  const msg: string = `Document with id ${document.id} is not payed. Make the payment to get additional info`;  console.log('--');  console.log(msg);  console.log('--');  }  }  payForDocument(id: number): void {  AccountingServiceProxied.accountingService.payForDocument(id);  }  } |

**Результати роботи програми**

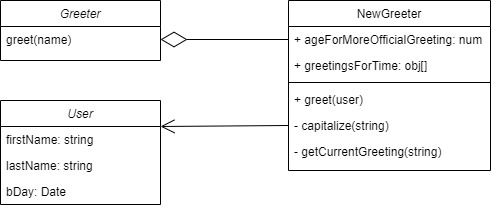


**Задача 2**

**Обґрунтування вибору шаблону**

Шаблон «Адаптер» використовується для підлаштування реалізації під новий необхідний інтерфейс. У нашому випадку необхідно замінити старий варіант системи вітання на новий, що бере до уваги вік користувача.

**Діаграма класів**



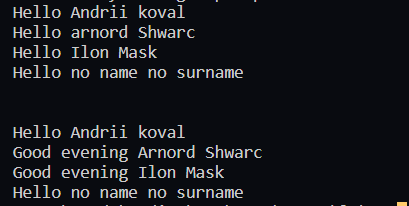
**Фрагменти коду програми**

|  |
| --- |
| **Стара система вітання** |
| export class Greeter {  static greet(name: string) {  console.log(`Hello ${name}`);  }  } |

|  |
| --- |
| **Нова система вітання** |
| import { User } from './User';  import { Greeter } from './Greeter';  export class NewGreeter {  public static ageForMoreOfficialGreeting: number = 35;  private static greetingsForTime = [  {  hour: {  from: 5, to: 11  },  message: 'Good morning'  }, {  hour: {  from: 11, to: 16  },  message: 'Good afternoon'  }, {  hour: {  from: 16, to: 22  },  message: 'Good evening'  }, {  hour: {  from: 22, to: 5  },  message: 'Good night'  }  ];  private static getCurrentGreeting(): string {  const currentHour = new Date().getHours();  const greeting: string = NewGreeter.greetingsForTime.filter(g => currentHour >= g.hour.from && currentHour < g.hour.to)[0]?.message;  return greeting;  }  public static greet(user: User) {  const age = new Date().getUTCFullYear() - user.bDay.getUTCFullYear();  if (age > this.ageForMoreOfficialGreeting) {  const msg: string = `${NewGreeter.getCurrentGreeting()} ${NewGreeter.capitalize(`${user.firstName} ${user.lastName}`)}`;  console.log(msg);  } else {  Greeter.greet(user.firstName + ' ' + user.lastName);  }  }  private static capitalize(string: string): string {  return string.split(' ').map(s => s[0].toUpperCase() + s.slice(1)).join(' ');  }  } |

|  |
| --- |
| **Користувач** |
| export class User {  constructor(public firstName: string,  public lastName: string,  public bDay: Date){}  } |

**Результати роботи програми**



**Висновки**

Я ознайомився з основними характеристиками шаблонів «Заступник», «Фасад», «Адаптер» та «Легковаговик», запам’ятав поширені ситуацій, коли використання цих шаблонів є доцільним, набув вмінь та навичок реалізації шаблонів під час створення програмного коду..