++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 № 3**

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

**тема “Реалізація породжуючих 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**

Ознаойомлення з основними характеристиками всіх породжуючих шаблонів («Прототипу», «Фабричного методу», «Абстрактної фабрики», «Одинака», «Будівельника»), 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. Гра у карти. За допомогою шаблона проектування створити колоду карт. Для цього використати відповідні прототипи гральних карт кожної масті. У створених карт змінювати тільки їх ранг. На екран вивести список отриманих таким чином карт.

2. Організувати виробництво грамот для лауреатів міжшкільного конкурсу, дипломів для дипломантів та грамот для всіх інших учасників на основі використання шаблону «Абстрактна фабрика». Список учасників конкурсу містить такі дані: школа та клас учня, а також місце яке він посів (або примітка про присудження диплому конкурсу). Всі створені грамоти повинні бути незалежними об’єктами. Поліграфічні фірми, які займаються виготовленням продукції такого типу, відрізняються кількістю кольорів, що використовуються при друці, якістю паперу та відповідно ціною на послуги. Забезпечити у програмі вибір фірми з виготовлення грамот у залежності від наявної суми грошей

**Задача 1**

**Обґрунтування вибору шаблону**

Шаблон «Прототип» використовується для клонування інсуючих об’єктів без їх явного інстанціювання. У нашому випадку є однотипні карти, отримувати які можно за допомогою відповідних параметрів таких як ранг та масть. Також, за умовою завдання було використано шаблон «Одинак» для CardProvider.

**Діаграма класів**

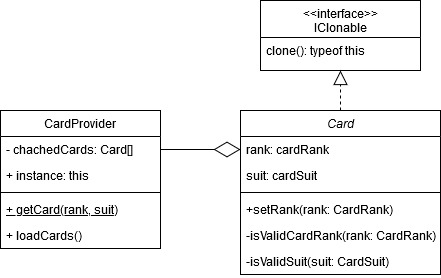


Рис. Діаграма класів завдання 1

**Фрагменти коду програми**

|  |
| --- |
| **Карта** |
| |  | | --- | |  |   **import** { ICloneable } from './icloneable';  **export** **enum** CardSuit {  SuitHeart, SuitDiamond, SuitClub, SuitSpade  }  **export** **enum** CardRank {  RankSix, RankSeven, RankEight, RankNine, RankTen,  RankJack, RankQueen, RankKing, RankAce  }  **export** **class** Card **implements** ICloneable {  **private** \_rank: **CardRank**;  **private** \_suit: **CardSuit**;  **constructor**({ rank, suit }: { rank: **CardRank**, suit: **CardSuit** }) {  **this**.\_rank = Card.isValidCardRank(rank) ? rank : **CardRank.RankSix**;  **this**.\_suit = Card.isValidSuit(suit) ? suit : **CardSuit.SuitHeart**;  }  **public** setRank(value: **CardRank**) {  **if** (Card.isValidCardRank(value)) {  **this**.\_rank = value;  } **else** {  **throw** **new** Error('Invalid card rank value');  }  }  get rank() {  **return** **this**.\_rank;  }  get suit() {  **return** **this**.\_suit;  }  clone() {  **return** **new** Card({ rank: **this.rank**, suit: **this.\_suit** }) **as** **this**;  }  **public** displayInfo() {  **const** msg =  ‘Card ----\n`+  `Rank: **$**{CardRank[**this**.\_rank]}\n`+  `Suite: **$**{CardSuit[**this**.\_suit]}`;  console.log(msg);  }  **private** **static** isValidCardRank(rank: **CardRank**) {  **return** Object.values(CardRank).includes(rank);  }  **private** **static** isValidSuit(suit: **CardSuit**) {  **return** Object.values(CardSuit).includes(suit);  }  **public** **static** getSuitEnumKeys() {  **return** Object.values(CardSuit).map(v => parseInt(v **as** **string**)).filter(v => !isNaN(v));  }  **public** **static** getRankEnumKeys() {  **return** Object.values(CardRank).map(v => parseInt(v **as** **string**)).filter(v => !isNaN(v));  }  } |

|  |
| --- |
| **ICloneable** |
| **export** **interface** ICloneable {  clone(): **this**;  } |
| **CardProvider** |
| **import** { Card, CardSuit, CardRank } from './card';  **export** **class** CardProvider {  **private** **static** \_instance: **CardProvider**;  **private** cachedCards: **Card**[] = [];  **public** getCard(rank: **CardRank**, suit: **CardSuit**) {  **const** card = **this**.cachedCards.find(c => c.rank === rank && c.suit === suit);  **if** (!card)  **return** **null**;  **return** card.clone();  }  **public** **static** get instance(): CardProvider {  **if** (!CardProvider.\_instance) {  CardProvider.\_instance = **new** CardProvider();  }  **return** CardProvider.\_instance;  }  **public** loadCards(): **void** {  **const** ranks = Card.getRankEnumKeys();  **const** suits = Card.getSuitEnumKeys();  **for** (**const** rank of ranks) {  **for** (**const** suit of suits) {  **this**.cachedCards.push(**new** Card({ rank, suit }));  }  }  // just for demo  // console.log(this.cachedCards, this.cachedCards.length);  }  } |

|  |
| --- |
| **main** |
| // …  CardProvider.instance.loadCards();  **function** sample1() {  **const** card1 = CardProvider.instance.getCard(CardRank.RankAce, CardSuit.SuitDiamond);  **const** possibleCopy = CardProvider.instance.getCard(CardRank.RankAce, CardSuit.SuitDiamond);  card1.setRank(CardRank.RankKing);  console.log(card1, possibleCopy);  }  sample1();  **function** sample2() {  **const** ranks = Card.getRankEnumKeys();  **const** suits = Card.getSuitEnumKeys();  **const** deck: **Card**[] = [];  **for** (**const** suit of suits) {  **for** (**const** rank of ranks) {  **const** card: **Card** = CardProvider.instance.getCard(rank, suit);  deck.push(card);  }  }  deck.forEach(c => c.displayInfo());  assert.equal(deck.length, **36**);  }  sample2(); |

**Результати роботи програми**

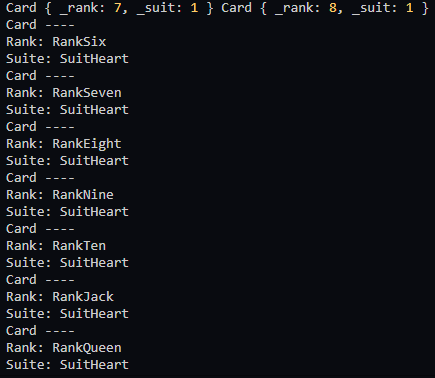


Рис. Частковий вивід програми

**Задача 2**

**Обґрунтування вибору шаблону**

Шаблон «Абстрактна фабрика» використовується для спрощеного строверння різних об’єктів за допомогою різних так званих фабрик, які мають єдиний інтерфейс. У нашому випадку є різні типи вітальних документів та різні види фабрик, які користувач обирає в залежності від ціни та палітри кольорів. Звідси у різних фабрик документи різної якості, але клієнтський код завжди має справу лише з абстрактною фабрикою, не вдаючись у подробиці її реалізації.

**Діаграма класів**

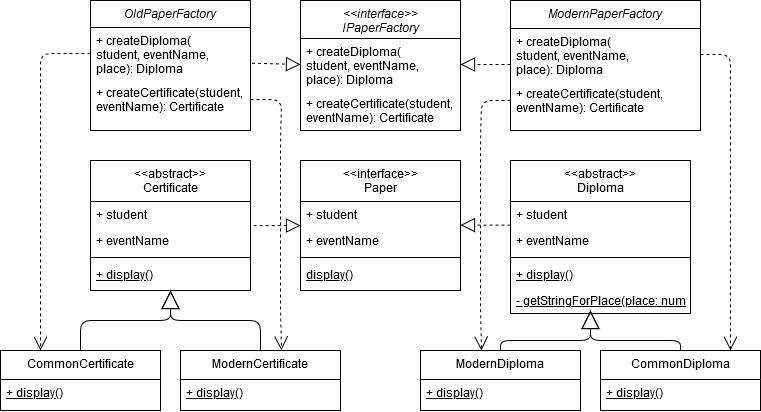


Рис. Діаграма класів завдання 2

**Фрагменти коду програми**

|  |
| --- |
| **Інтерфейс асбтрактнох фабрики** |
| **export** **interface** IPaperFactory {  createDiploma(student: **Student**, eventName: **string**, place: **number**): Diploma;  createCertificate(student: **Student**, eventName: **string**): Certificate;  } |

|  |
| --- |
| **Дві конкретні реалізації фабрик** |
| **export** **class** ModernPaperFactory **implements** IPaperFactory {  createDiploma(student: **Student**, eventName: **string**, place: **number**): Diploma {  console.log('Creating diploma with best materials and colours!');  **const** diploma = **new** ModernDiploma(student, eventName, place);  **return** diploma;  }  createCertificate(student: **Student**, eventName: **string**): Certificate {  console.log('Creating certificate with best materials and colours!');  **const** certificate = **new** ModernCertificate(student, eventName);  **return** certificate;  }  }  **export** **class** OldPaperFactory **implements** IPaperFactory {  createDiploma(student: **Student**, eventName: **string**, place: **number**): Diploma {  console.log('Creating diploma with in usuall factory');  **const** diploma = **new** CommonDiploma(student, eventName, place);  **return** diploma;  }  createCertificate(student: **Student**, eventName: **string**): Certificate {  console.log('Creating certificate in usuall factory');  **const** certificate = **new** CommonCertificate(student, eventName);  **return** certificate;  }  } |

|  |
| --- |
| **Різні види документів** |
| **import** { Certificate, Diploma } from "./abstractPapers";  **export** **class** CommonCertificate **extends** Certificate {  display() {  **super**.display();  console.log('Common sertificate made with cheap materials');  }  }  **export** **class** CommonDiploma **extends** Diploma {  display() {  **super**.display();  console.log('Common diploma made with cheap materials');  }  }  **export** **class** ModernCertificate **extends** Certificate {  display() {  **super**.display();  console.log('Modern sertificate made with hi-end materials');  }  }  **export** **class** ModernDiploma **extends** Diploma {  display() {  **super**.display();  console.log('Modern diploma made with hi-end materials');  }  } |

|  |
| --- |
| **Клієнтський код** |
| **function** getFactoryFromSpecs(moneyAvailable: **number**, coloursNeeded: **number**): IPaperFactory {    **const** factorySpecs = [modernFactorySpecs, oldFactorySpecs]  .find( ({ colours, price }) => colours >= coloursNeeded && price <= moneyAvailable);  **if** (!factorySpecs)  **return** **null**;  **switch** (factorySpecs) {  **case** oldFactorySpecs: **return** **new** OldPaperFactory();  **case** modernFactorySpecs: **return** **new** ModernPaperFactory();  }  **return** **null**;  }  (**function** main() {  **const** moneyAvailable = args.m || args.money;  **const** coloursNeeded = args.c || args.colours;  **const** factory = getFactoryFromSpecs(moneyAvailable, coloursNeeded);    **if** (!factory) {  console.log('Sorry, no factory matches your possiblitities');  **return**;  }  **const** student = **new** Student('andrii', 'koval', 'kpi', 'kp83');  **const** diploma = factory.createDiploma(student, 'int20h', **2**);  **const** certificate = factory.createCertificate(student, 'math olymp');  diploma.display();  certificate.display();  })(); |

**Результати роботи програми**

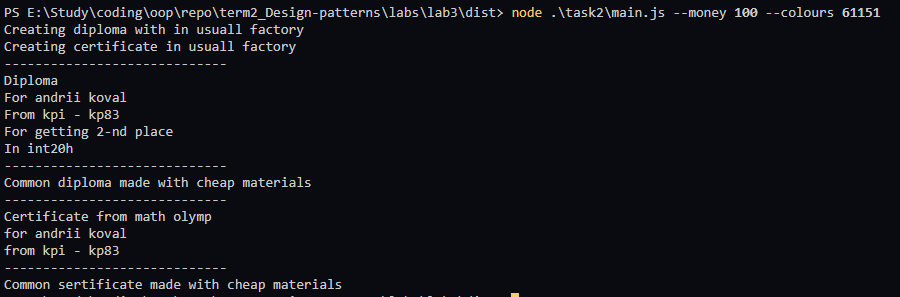
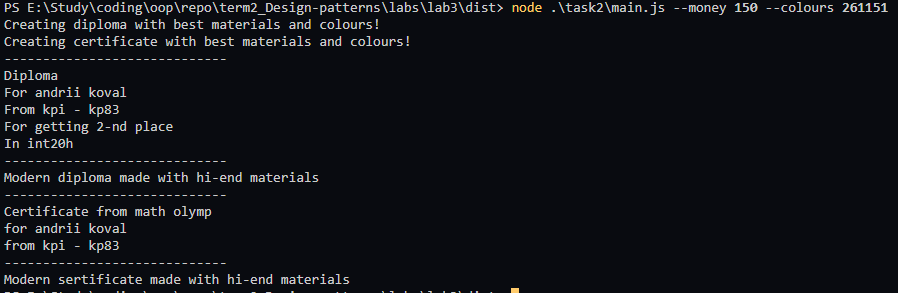


Рис. Приклад роботи з різними вимогами до друку

**Висновки**

Я ознайомився з основними характеристиками всіх породжуючих шаблонів («Прототипу», «Фабричного методу», «Абстрактної фабрики», «Одинака», «Будівельника»), запам’ятав поширені ситуації, коли використання цих шаблонів є доцільним, набув вмінь та навичок реалізації шаблонів під час створення програмного коду.