НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматизованих систем обробки інформації і управління

**КУРСОВА РОБОТА**

з дисципліни: «Основи програмування - 2. Основи об'єктно-орієнтованого програмування»

на тему: «Карткова гра Скарбнички»

Студента 1 курсу, групи ІП-02

напряму підготовки «ІПЗ»

спеціальності 121

Василенка П. О.

Керівник ст. вик. Головченко М. М.

Національна оцінка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії: |  |  | ст. вик. Головченко М. М. |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  | к. т. н. доц. Муха І. П. |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2021 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-02 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

**Василенка Павло Олександровича**

(прізвище, ім’я, по батькові)

1. Тема роботи: “Карткова гра Скарбнички”

2. Строк здачі студентом закінченої роботи: «04» червня 2021 р.

3. Вихідні дані до роботи: Технічне завдання (Додаток А)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці): Вступ, постановка задачі, теоретичні відомості, опис алгоритмів, опис архітектури програмної системи, тестування програмного забезпечення, інструкція користувача, висновок, перелік посилань.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання: 08.03.2021

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 08.03.2021 |  |
| 2. | Підготовка технічного завдання | 10.03.2021 |  |
| 3. | Аналіз предметної області | 15.03.2021 |  |
| 4. | Проектування архітектури програмної системи | 29.03.2021 |  |
| 5. | Узгодження з керівником інтерфейсу користувача | 31.03.2021 |  |
| 6. | Розробка сценарію роботи програми | 05.04.2021 |  |
| 7. | Розробка програмного забезпечення | 26.04.2021 |  |
| 8. | Узгодження з керівником плану тестування | 28.04.2021 |  |
| 9. | Тестування програми | 03.05.2021 |  |
| 10. | Підготовка пояснювальної записки | 20.05.2021 |  |
| 11. | Здача курсової роботи на перевірку | 02.06.2021 |  |
| 12. | Захист курсової роботи | 04.06.2021 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Василенко П. О. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |
|  |  |  |  |
| Керівник |  |  | Головченко М.М. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

" " 2021 р

Анотація

Пояснювальна записка до курсової роботи: **xx** сторінок, 16 рисунків, 14 таблиць, 2 формули, 2 посилання.

Об’єкт дослідження: “Карткова гра Скарбничка”.

Метою курсової роботи є розробка програмного забезпечення, щопередбачає можливість гри в «Скарбнички» проти комп’ютера.

В процесі розробки було сконструйовано власний алгоритм прийому кращих ходів для комп’ютерного опонент. Приведені змістовні постановки задач та виконано усі задачі з переліку.

“СКАРБНИЧКИ”, КАРТКОВА ГРА, АЛГОРИТМ ПРИЙОМУ РІШЕНЬ, ВЕБ-ДОДАТОК.

# Зміст

[Зміст 4](#_Toc73352229)

[Вступ 6](#_Toc73352230)

[1 Постановка задачі 7](#_Toc73352231)

[2 Теоретичні відомості 8](#_Toc73352232)

[3 Опис алгоритмів 9](#_Toc73352233)

[3.1 Загальний алгоритм комп’ютера під час ходу гравця 9](#_Toc73352234)

[3.2 Загальний алгоритм ходу комп'ютера 10](#_Toc73352235)

[3.3 Загальний алгоритм вибору кращого ходу 13](#_Toc73352236)

[3.4 Загальний алгоритм дій, коли один з гравців збирає скарбничку 15](#_Toc73352237)

[4 Опис архітектури програмної системи 16](#_Toc73352238)

[4.1 UML діаграми 16](#_Toc73352239)

[4.2 Таблиця методів 19](#_Toc73352240)

[5 Тестування програмного забезпечення 32](#_Toc73352241)

[5.1 План тестування 32](#_Toc73352242)

[5.2 Приклади тестування 32](#_Toc73352243)

[6 Інструкція користувача 33](#_Toc73352244)

[6.1 Робота з програмою 33](#_Toc73352245)

[6.2 Формат вхідних та вихідних даних 33](#_Toc73352246)

[6.3 Системні вимоги 33](#_Toc73352247)

[Висновок 34](#_Toc73352248)

[Перелік посилань 35](#_Toc73352249)

[Додаток А Технічне завдання 36](#_Toc73352250)

[Додаток Б. Тексти програмного коду 39](#_Toc73352251)

# Вступ

З розвитком індустрії інформаційних технологій збільшується попит на ігри. Так, збільшується і кількість роботів, що вміють грати в ці ігри. Сьогодні складно собі уявити гру в якій не буде штучного інтелекту.

Люди створюють штучний інтелект, що зможе перевершити їх. Так у 1997 році було створено Deep Blue, комп’ютера, що зміг перемогти світового чемпіона в шахах Гаррі Каспарова.

Більш складною задачею виявилася розробка штучного інтелекту для перемоги в китайську древню гру «Го». Лише у 2015 році програма AlphaGo, розроблена компанією Deepmind перемогла трьохкратного чемпіона Європи Фань Хуея з рахунком 4-1.

Усі перераховані вище роботи, що змогли перемогти людину використовували машинне навчання(machine learning), що давало їм самостійно вчитися та запам’ятовувати кращі ходи. Метою цієї роботи є розробка карткової гри «Скарбнички» проти комп’ютерного інтелекту, з заздалегідь продуматим лінійним алгоритмом. Для правильного алгоритму гри має бути проаналізовано всі можливі ситуації та виходи з них.

Шляхом практичних досліджень було встановлено всі можливі випадки в грі та те, як має діяти людина в ідеальному випадку для перемоги. Перевагою людини є зчитування емоцій, що допомагає в прийнятті рішень. Комп’ютер же має підходити до всього лише математично.

# Постановка задачі

Розробити програмне забезпечення, що дає змогу користувачу грати в карткову гру «Скарбнички» проти комп’ютерного суперника.

Відкривши чи оновивши сторінку веб-застосунку перед користувачем з’явиться модальне вікно з кнопкою початку гри. Натиснувши на цю кнопку розпочнеться гра за відповідними правилами описаними в розділі «Теоретичні відомості». Правила порушувати неможливо. Інтерфейс користувача включає в себе можливість слідкувати за кількістю карт в колоді, в руці суперника, кількістю скарбничок обох гравців, та картами в своїй руці. По закінченню гри має виводитися вікно з оголошенням перемоги/поразки гравця з кнопкою «Грати знову».

# Теоретичні відомості

Правила гри «Скарбнички»:

Кількість колод: 1

Кількість карт в колоді: 36

Кількість гравців: 2

Старшинство карт: 6,7,8,9,10,J,Q,K,A

Ціль гри: зібрати якнайбільше скарбничок, до того моменту, як карти в колоді й у обох гравців закінчаться

Хід гри: Колода ретельно тасується. Кожен гравець отримує по 4 карти, а колода, що залишилася, кладеться на центр столу сорочкою догори. Першим ходить користувач – надалі по черзі. Спочатку гравець питає у суперника про наявність карти/карт таким чином: значення карти, якщо вгадує, то вгадує кількість карт і якщо угадує то питає масті. Гравець може спитати лише карту такого значення, яке є в нього в руці. Якщо у суперника наявні всі запитані карти – він віддає їх гравцю. Якщо у суперника не виявилося жодної з запитаних карт – гравець бере верхню карту з колоди. Якщо у суперника є лише частина з загаданих карт, то він віддає їх гравцю, а гравець бере карту з колоди. Якщо у гравця чи комп’ютера в руках виявиться 4 однакові за значенням карти – вони збираються в скарбничку й відповідний гравець прибирає їх зі своєї руки. Якщо на будь-якому етапі гри у гравця виявилося менше 4-х карт – він бере з колоди рівно стільки карт, скільки йому не вистачає. Гру вважають закінченою, коли колода карт – пуста а у гравців нема карт в руках. Після цього рахуються скарбнички кожного та визначається переможець.

# Опис алгоритмів

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| enemyHand | Рука комп’ютера – масив карт, які комп’ютер може використовувати |
| deckSize | Кількість карт, що залишилися в колоді (від 0 до 28) |
| imagineHand | Уявна рука гравця – комп’ютер запам’ятовує всі карти, що гравець в нього забирав, угадуючи, а також всі карти, що гравець питав, але не вгадував |
| lastUsed | Масив карт, що комп’ютер використовував нещодавно, тому немає сенсу використовувати ще раз. Максимальна к-сть таких карт – 3. |
| value | Значення карти – 6 7 8 9 10 J Q K A |
| suit | Масть карти - ♣ ♠ ♦ ♥ |
| chestsArr | Масив скарбничок, що є кожного гравця |
| chestsNum | Кількість скарбничок у кожного гравця |

## Загальний алгоритм комп’ютера під час ходу гравця

1. ПОЧАТОК
2. Якщо deckSize>0 і enemyHand.size>0
   1. Гравець запитує value карти
   2. Якщо гравець вгадав
      1. Гравець запитує кількість таких карт
      2. Якщо гравець вгадав
         1. Гравець вгадує масті карт
         2. Якщо гравець вгадав усі карти
            1. Передати вгадані гравцем карти
            2. Записати value, accurate=true, isNot=[], suit до ImagineHand
            3. Записати value, accurate=false, до ImagineHand
            4. Почати хід (алгоритм 3.2)
         3. Інакше якщо гравець вгадав частину карт
            1. Передати вгадані гравцем карти

Продовження алгоритму 3.1

* + - * 1. Записати value, accurate=true, isNot = [], suit до ImagineHand (всі карти, що гравець вгадав)
        2. Додати value, accurate=false,isNot=[suit] до ImagineHand (карти що гравець назвав та не вгадав)
        3. Почати хід (алгоритм 3.2)
      1. Інакше
         1. Записати value, accurate=false, isNot = [suit] до ImagineHand (всі масті названі гравцем)
         2. Почати хід (алгоритм 3.2)
    1. Інакше
       1. Записати value, accurate=false до ImagineHand, якщо такої ще нема
       2. Почати хід (алгоритм 3.2)
  1. Інакше
     1. Записати value, accurate=false до ImagineHand, якщо такої ще нема
     2. Почати хід (алгоритм 3.2)

1. Інакше
   1. Гру завершено
2. КІНЕЦЬ

## Загальний алгоритм ходу комп'ютера

1. ПОЧАТОК
2. Поки enemyHand.size<4
   1. Взяти карту з колоди
3. Кінець поки
4. Priority = -1, bestcardValue = null
5. Якщо в deckSize==0
   1. bestcardValue = першій в руці карті
6. Інакше
   1. Для кожної можливої карти з масиву values
      1. Визначити newPriority цієї карти
      2. Якщо newPriority > priority
         1. Priority=newPriority
         2. bestCardValue = value

Продовження алгоритму 3.2

1. Якщо обрана карта є у гравця
   1. Якщо deckSize=0
      1. numberToAsk = 4-(кількість цієї карти в руці комп’ютера)
   2. Інакше
      1. numberToAsk = кількість цієї карти в ImagineHand
      2. Якщо numberToAsk=0
         1. numberToAsk++
   3. inHand = кількість карт з value == bestCardValue в руці комп’ютера
   4. Поки (inHand+numberToAsk>4)
      1. numberToAsk—
   5. Якщо вгадали кількість карт в руці гравця
      1. В imagineHand знайти карти з value==bestCardValue і accurate == true, і додаємо їх до suitsInImagineAccurate
      2. Знайти всі карти з value==bestCardValue і isNot.length>0, і кожну масть з масивів isNot.length записуємо до suitsInImagineNotArr
      3. Знайти всі карти в руці з value==bestCardValue, та записуємо масті до suitsInHand
      4. Враховуючи інформацію з попередніх трьох пунктів створити suitsToAsk (обов'язково мають бути масті з suitsInImagineAccurateб і не має бути мастей з suitsInHand та suitsInImagineNotArr)
      5. Для кожної карти, що намагаємося вгадати
         1. Якщо ця карта є у гравця
            1. Забрати карту
            2. Покласти собі в руку
            3. Видалити цю карту з imagineHand(з найменшою кількістю інформації)
         2. Інакше
            1. Якщо в ImagineHand ще нема карти з таким value, accurate=false, isNot.length=0

Додати таку карту

* + - * 1. Інакше

Продовження алгоритму 3.2

Не додавати нічого

* + 1. Кінець циклу
    2. Якщо к-сть вгаданих карт – 0
       1. Узяти карту з колоди
       2. Перевірити на скарбничку (адгоритм 3.4)
    3. Інакше якщо кількість вгаданих карт менша за кількість вгадуваних карт
       1. Узяти карту з колоди
       2. Перевірити на скарбничку
    4. Інакше
       1. Нічого не робити
    5. Перевірити на наявність карти в масиві скарбничок
    6. Якщо вона там є
       1. Прибрати всі копії цієї карти з ImagineHand і LastUsed
  1. Інакше
     1. Якщо в ImagineHand 1 копія карти
        1. Додати value=bestCardValue accurate = false isNow=[]
     2. Інакше якщо 3 карти АБО 2 карти і deckSize>14
        1. Видалити одну копію карти з ImagineHand(з найменшою кількістю інформації)
     3. Інакше якщо 0 карт
        1. Додати value=bestCardValue accurate = false isNow=[]. Двічі.
     4. Інакше якщо 2 карти
        1. Додати value=bestCardValue accurate = false isNow=[].
     5. Взяти карту з колоди
     6. Перевірити на наявність скарбничок (алгоритм 3.4)

1. Інакше
   1. Взяти карту з колоди
   2. Перевірити на наявність скарбничок (алгоритм 3.4)
2. Поки enemyHand.size<4
   1. Взяти карту з колоди
   2. Перевірити на наявність скарбнички (алгоритм 3.4)
3. КІНЕЦЬ

## Загальний алгоритм вибору кращого ходу

1. ПОЧАТОК
2. newPriority=0;
3. Визначити кількість цієї карти в руці. (inHand)
4. Якщо inHand==0
   1. NewPriority=-1
5. Інакше якщо inHand==1
   1. Визначити наявність цієї карти в lastused (cardInLastUsed)
   2. Якщо ця карта є в lastUsed
      1. Порахувати кількість точних (accurate==true) мастей в ImagineHand
      2. Якщо їх 0
         1. newPriority = 0
      3. Інакше якщо їх 1
         1. newPriority = 1200
      4. Інакше якщо їх 2
         1. newPriority = 2500
      5. Інакше якщо їх 3
         1. newPriority = 10000
   3. Інакше
      1. Порахувати кількість цієї карти в ImagineHand
         1. Якщо їх 0
            1. newPriority = 1000
         2. Якщо їх 1
            1. Якщо це точна масть

newPriority = 1800

* + - * 1. Інакше

newPriority = 1100

* + - 1. Якщо їх 2
         1. Якщо точних мастей серед них 0

newPriority = 2100

* + - * 1. Інакше якщо точних мастей 1

newPriority = 2500

* + - * 1. Інакше якщо точних мастей 2

newPriority = 2800

* + - 1. Якщо їх 3

Продовження алгоритму 3.3

* + - * 1. newPriority = 10000

1. Інакше якщо inHand==2
   1. Визначити чи є ця карта в lastused(cardInLastUsed)
   2. Якщо вона там є
      1. Визначити кількість точних мастей в ImagineHand
      2. Якщо їх 0
         1. newPriority = 0
      3. Інакше якщо їх 1
         1. newPriority = 2800
      4. Інакше якщо їх 2
         1. newPriority = 10000
   3. Інакше
      1. Визначити кількість цієї карти в ImagineHand
      2. Якщо їх 0
         1. newPriority = 1800
      3. Інакше якщо їх 1
         1. Якщо це точна масть
            1. newPriority = 2800
         2. Інакше
            1. newPriority = 2500
      4. Інакше якщо їх 2
         1. newPriority = 10000
2. Інакше якщо inHand==3
   1. Визначити чи є ця карта в lastused(cardInLastUsed)
   2. Якщо вона там є
      1. Визначити кількість цієї карти в ImagineHand
      2. Якщо їх 0
         1. newPriority = 0
      3. Інакше якщо їх 1
         1. newPriority = 10000
   3. Інакше
      1. Визначити кількість цієї карти в ImagineHand
      2. Якщо їх там 0
         1. newPriority = 2600
      3. Якщо їх там 1

Продовження алгоритму 3.3

* + - 1. newPriority = 10000

1. Інакше якщо inHand==4
   1. Перевірити на наявність скарбнички (алгоритм 3.4)
2. КІНЕЦЬ

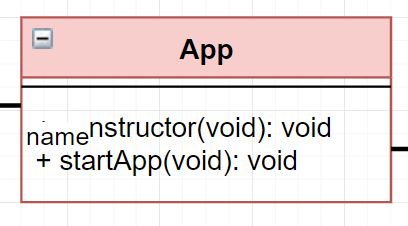
## Загальний алгоритм дій, коли один з гравців збирає скарбничку

1. ПОЧАТОК
2. Counter = 0
3. Для кожної карти в руці
   1. Якщо її value співпадає з шуканим
      1. Counter++
4. Якщо counter==4
   1. Прибрати всі карти з цим value з руки, imagineHand, LastUsed
   2. Додати цей value до chestsArr
   3. chestsNum++
5. Якщо після збоору скарбнички deckSize==0 і enemyHand.size == 0
   1. Порахувати скарбнички кожного гравця
   2. Оголосити переможця
6. КІНЕЦЬ

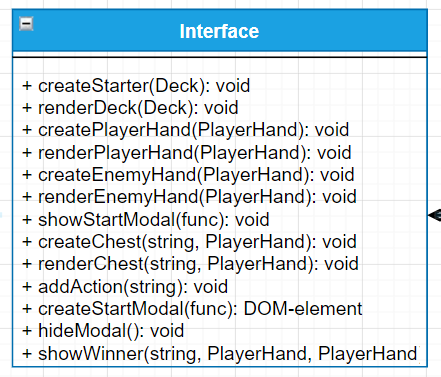
# Опис архітектури програмної системи

Опис програмного продукту представлений у двох форматах: у вигляді UML діаграм (Рисунок 4.1 — 4.10) та таблиці методів (Таблиця 4.1 — 4.2).

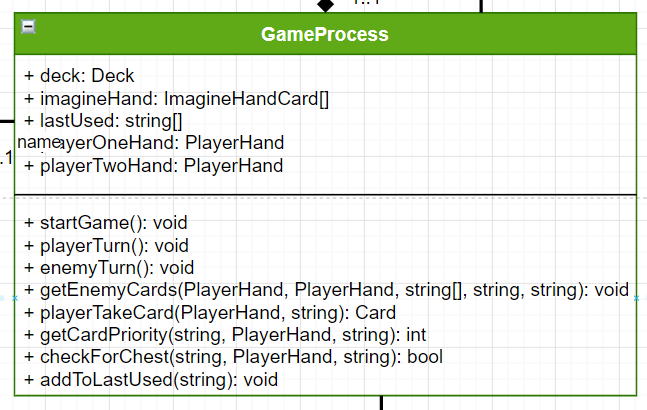
## UML діаграми



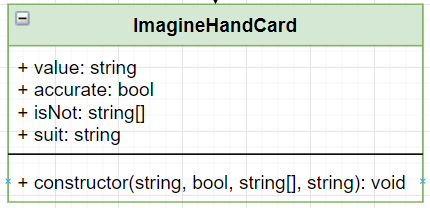
* + - * 1. Клас «App»



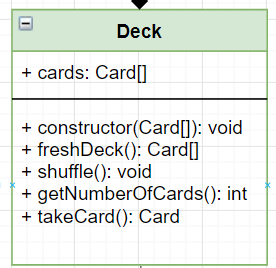
* + - * 1. Клас «Interface»



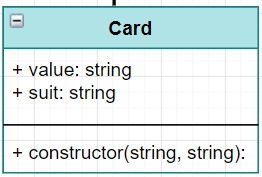
* + - * 1. Клас «GameProcess»



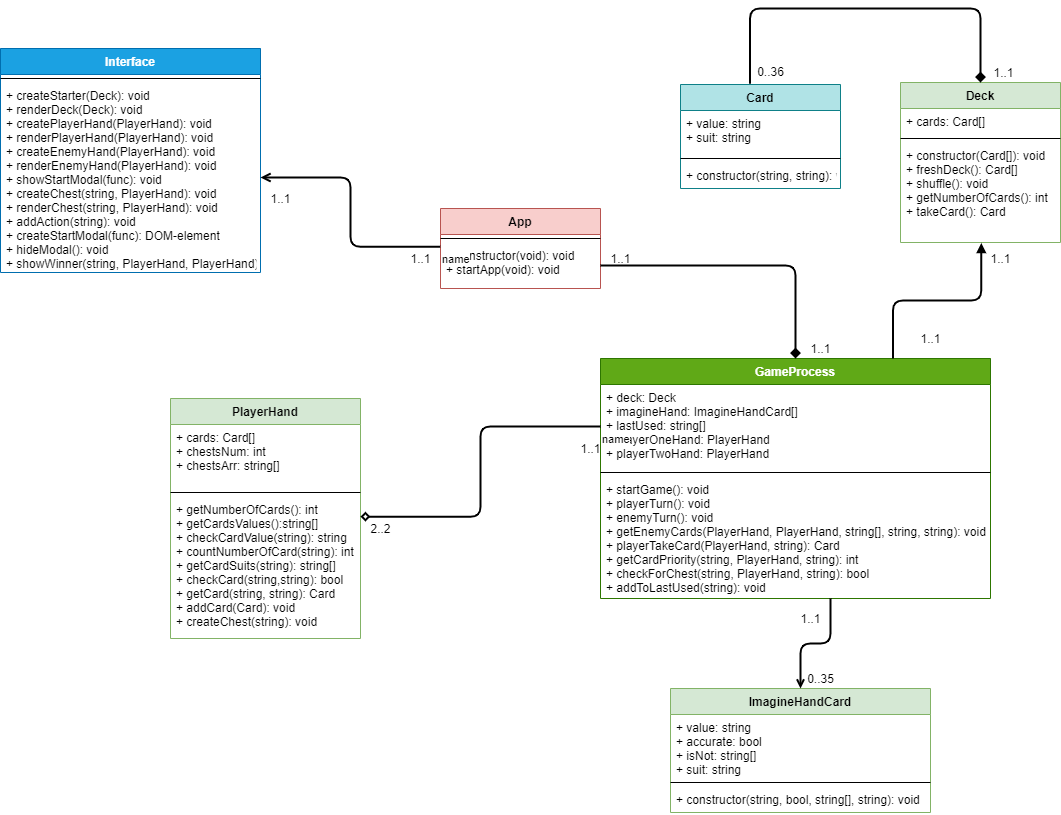
* + - * 1. Клас «ImagineHandCard»



* + - * 1. Клас «Deck»



* + - * 1. Клас «Card»



* + - * 1. Загальна діаграма класів

## Таблиця методів

У таблиці 4.1 представлено вбудовані методи

Вбудовані методи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | Array | push | Вставка нового елемента в масив | Будь-який | Довжина новоутвореного масиву | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 2 | Array | map | Створення нового масиву з результатами виклику вказаної функції для кожного елементу масиву | Функція, яку буде виконано до кожного елемента масиву | Новий масив, де кожен елемент це результат виклику функції | - |
| 3 | Array | pop | Вилучення останнього елементу масива | - | Видалений елемент | - |
| 4 | Node | append | Додати вузли в DOM-дерево | Елементи DOM-дерева | - | - |
| 5 | Node | remove | Видалити вузол з DOM-дерева | - | - | - |
| 6 | Node | appendChild | Додати вузол до DOM-дерева | - | - | - |
| 7 | Node | setAttribute | Додати властивість до елемента DOM-дерева | Перший параметр – атрибут, що треба встановити. Другий – значення цього атрибута | - | - |
| 8 | Node | createElement | Створення нового елемента DOM-дерева | DOM-string тега, що треба створити | Створений елемент | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 9 | String | split | Розбиття рядка на кілька елементів для отримання масиву | Символ-роздільник | Новий масив | - |
| 10 | Array | filter | Отримання нового масиву з елементів попереднього масиву, що задовольняють вказану умову | Функція , за якою проводить перевірка | Новий масив | - |
| 11 | Array | forEach | Виконання одній і тієї самої функції для кожного елементу масива | Функція, яку має бути виконано | - | - |
| 12 | Object | keys | Отримати всі ключі об’єкта | Об'єкт | Масив ключів | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 13 | Array | find | Знайти елемент, що задовольняє умові, вказаній у функції. Відрізняється від filter тим, що при знаходженні першого задовольняючого елемента повертає його і завершує роботу | Функція, за якою відбувається перевірка | Перший знайдений елемент або undefined | - |
| 14 | Array | findIndex | Знайти елемент, що задовольняє умові, вказаній у функції.Повернути його індекс | Функція, за якою відбувається перевірка | Індекс знайденого елемента або -1 | - |
| 15 | DOM-об’єкт | querySelector | Всередині елемента, що вказано знайти DOM-елемент за вказаною DOM-string | DOM-string | Знайдений елемент або null | - |
| 16 | DOM-об’єкт | getElementsByClassName | Отримати колекцію з усіх елементів всередині вказаного елементу DOM, що мають вказаний клас | Рядок з вказаним класом | Колекція елементів DOM | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 17 | Array | getElementById | Отримати елемент усередині вказаного елементу DOM, що має вказаний id | Рядок з вказаним id | DOM-елемент | - |
| 18 | Таргет-об’єкт | addEventListener | Поставити на певний елемент подію, що буде відбуватися за певних умов | 1)Подія 2)функція, яку буде виконано при спрцюванні події | - | - |
| 19 | Location | reload | Перезавантаження сторінки | - | - | - |
| 20 | Array | includes | Перевірка на наявність вказаного елемента в масиві | Елемент, який треба перевірити | True || false | - |
| 21 | Array | from | Створення з елемента будь-якого типу масив | Елемент, що треба перетворити | Отриманий масив | - |
| 22 | Math | floor | Округлення числа вниз | Число, яке треба округлити | Округлене число | - |
| 23 | Set | add | Додати елемент до Set() | Елемент який треба додати | - | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 24 | Array | shift | Видалення першого елемента з масива | - | Видалений елемент | - |

У таблиці 4.2 представлено користувацькі методи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | App | constructor | Конструктор класу. Запускає роботу додаткушляхом запуску методу цього ж класу - startApp | - | - | - |
| 2 | App | startApp | Метод, що запускає роботу всього додатку. Створює колоду карт, перемішує її, створює руки карт обох гравців, виводить стартове модальне вікно, у яке передає параметри закриття, а саме: функції, що | - | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | App | startApp | мають бути викликані: створення початкового поля, заповнення картами рук гравців, створення на полі скарбничок, та свторення екземпляру класу GameProcess, чим запускає його конструктор |  |  |  |
| 3 | Deck | constructor | Конструктор класу. Створює колоду з 36 впорядкованих карт після чого перемішує їх | Отримує масив карт, що генерується методом freshDeck | - | - |
| 4 | Deck | freshDeck | З константних масивів VALUES і SUITS створює всі можливі комбінації та записує в масив. | - | Масив з 36 впорядкованих карт | - |
| 5 | Deck | Get numberOfCards | Повертає кількість карт в колоді | - | Ціле число, що дорівнює кількості карт в колоді | - |
| 6 | Deck | shuffle | Перемішування колоди. Кожна карта випадковим чином міняється місцями з іншою випадковою картою. | - | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | Deck | takeCard | Взяти карту з колоди, якщо вона не пуста | - | Верхня карта з колоди або undefined | - |
| 8 | PlayerHand | Constructor | Конструктор класу. Створює пустий масив карт, Пустий масив скарбничок, змінну для підрахунку скарбничок та бере 4 карти з колоди використовуючи метод takeCard з колоди | deck | - | - |
| 9 | PlayerHand | Get numberOfCards | Визначає кількість карт в руці | - | Кількість карт в руці, ціле число | - |
| 10 | PlayerHand | Get cardValues | Дає можливість отримати масив зі значень карт в руці | - | Масив рядків, значень карт | - |
| 11 | PlayerHand | checkCardValue | Перевіряє чи є карта з вказаним value в руці | Value – рядок зі значенням карти | Повертає знайдену карту чи undefined | - |
| 12 | PlayerHand | CountNumberOfCard | Визначає кількість карт з вказаним value в руці | Value – рядок зі значенням карти | Ціле число, кількість карт з value в руці | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 13 | PlayerHand | getCardSuits | Отримати всі масті карти з вказаним value , що є в руці | Value – рядок зі значенням карти | Масив рядків – мастей | - |
| 14 | PlayerHand | CheckCard | Перевірка на наявність карти з вказаним value і suit в руці | Value – рядок зі значенням карти Suit – рядок зі значенням масті карти | True або false | - |
| 15 | PlayerHand | getCard | Забрати карту з вказаними value і suit у гравця | Value – рядок зі значенням карти Suit – рядок зі значенням масті карти | Вирізана карта, екземпляр класу Card | - |
| 16 | PlayerHand | addCard | Покласти карту собі в руку | Card | - | - |
| 17 | PlayerHand | createChest | Видаляє з руки всі карти з вказаним value та додає їх у скарбничку гравця, що викликав функцію | Value – рядок зі значенням карти | - | - |
| 18 | Interface | createStarter | Створює на ігровому полі колоду карт а також поле з історією гри | Deck – колода карт | - | - |
| 19 | Interface | renderDeck | Якщо в колоді є карти, покаже скільки їх залишилося. Якщо в колоді карт нема – видалить колоду з поля | Deck | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20 | Interface | createPlayerHand | Створює руку гравця на полі та заповнює її картами | playerHand – рука гравця | - | - |
| 21 | Interface | renderPlayerHand | Оновлює карти в руці гравця | playerHand – рука гравця | - | - |
| 22 | Interface | createEnemyHand | Створює руку комп'ютера на полі та заповнює її картами | playerHand – рука комп’ютера | - | - |
| 23 | Interface | renderEnemyHand | Оновлює карти в руці комп'ютера | playerHand – рука комп’ютера | - | - |
| 24 | Interface | showStartModal | Створює модальне вікно. При закритті цього вікна викличеться функція onClose | onClose – функція, що має викликатися при закритті | - | - |
| 25 | Interface | createChest | Створює на полі гри блок зі скарбничками. | Player – рядок що вказує на номер гравця | - | - |
| 26 | Interface | renderChest | Оновлює кількість скарбничок у вказаного гравця | Player – рядок що вказує на номер гравця playerHand – рука цього гравця | - | - |
| 27 | Interface | addAction | Додати повідомлення message до ігрової історії | message | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 28 | Interface | createStartModal | Створює DOM-елемент модального вікна | onClose – функція, що має бути викликана при закритті модального вікна | DOM-елемент | - |
| 29 | Interface | hideModal | Приховати модальне вікно | - | - | - |
| 30 | Interface | showWinner | Вивести на екран переможця та скарбнички обох гравців | Player – рядок що вказує на номер гравця playerOneHand – рука першого гравця playerTwoHand – рука другого гравця | - | - |
| 31 | GameProcess | Constructor | Конструктор класу. Створює масиви imagineHand & lastUsed. Запам’ятовує deck, руки гравців і запускає гру | playerOneHand, playerTwoHand, deck | - | - |
| 32 | GameProcess | startGame | Запускає гру, перший ходить гравець | - | - | - |
| 33 | GameProcess | playerTurn | Метод виводить варіанти для ходу гравця, робить потрібні перевірки а в кінці ходу передає хід супернику | - | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 34 | GameProcess | enemyTurn | Метод відповідає за хід комп’ютера. Робить необзідні перевірки, а в кінці передає хід гравцеві | - | - | - |
| 35 | GameProcess | getEnemyCards | Відповідає за передачу карт між гравцями. Перевіряє їх наявність, передає, викликає перевірки на скарбнички. А також доповнює логіку комп’ютера через додавання карт до imagineHand | ToHand – рука в яку треба класти карти FromHand – рука звідки брати карти suits- масив мастей value – значення карти player – номер гравця | - | - |
| 36 | GameProcess | playerTakeCard | Відповідний гравець бере карту, повідомення про це виводиться на екран | playerHand – в цю руку потрібно класти карту. player – номер гравця | Значення взятої карти | - |
| 37 | GameProcess | getCardPriority | Обирає кращий хід для комп’ютера. Детальніше в [алгоритмі 3.1](#_Загальний_алгоритм_вибору) | Value – карту, що перевіряємо зараз | Int – новий пріоритет | - |
| 38 | GameProcess | checkForChest | Перевіряє карту з вказаним value на скарбничку. Детальніше в [алгоритмі 3.4](#_Загальний_алгоритм_дій,) | Value – значення карти, яке перевіряється playerHand – рука, в якій перевіряємо на скарбничку player – номер гравця | True || false | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 39 | GameProcess | addToLastUsed | Тримає в пам’яті три останні карти якими ходив комп’ютер | Value – карта яку треба записати | - | - |

# Тестування програмного забезпечення

## План тестування

## Приклади тестування

# Інструкція користувача

## Робота з програмою

## Формат вхідних та вихідних даних

## Системні вимоги

# Висновок

# Перелік посилань

# Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко М. М.

«10» березня 2021 р.

Виконавець:

Студент Василенко П. О.

«10» березня 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Скарбнички»

з дисципліни:

«Основи програмування-2. Модульне програмування»

Київ 2021

* 1. *Мета*: Метою курсової роботи є розробка програмного забезпечення, що передбачає можливість гри в «Скарбнички» проти комп’ютера.
  2. *Дата початку роботи*: «02»березня 2021 р.
  3. *Дата закінчення роботи*: «01»червня 2021 р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* Можливість грати з комп’ютерним опонентом
* Можливість випадкової генерації колоди карт
* Можливість дотримування правил гри:
* 1. Можливість вгадати карти суперника, й забрати їх собі
* 2.Можливість вгадати лише частину набору карт суперника
* 3.Можливість не вгадати жодної з карт суперника
* 4.Можливість збирати 4 однакові за значенням карти в «Скарбницю»
* 5.Можливість перемоги/поразки відповідно до кількості скарбничок гравців

1. Нефункціональні вимоги:

* Використання стандарту ES6 для Javascript
* Використання системи контролю версій Git
* Наявність зрозумілого користувачеві інтерфейсу
* Кросбраузерність (Google Chrome, Opera, Mozilla Firefox)
* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до 20.03.2021 р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до 02.04.2021р.)
3. Розробка програмного забезпечення (до 24.04.2021р.)
4. Тестування розробленої програми (до 01.05.2021р.)
5. Розробка пояснювальної записки (до 15.05.2021 р.).
6. Захист курсової роботи (до 04.06.2021 р.).
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

# Додаток Б. Тексти програмного коду

(Найменування програми (документа))

*Тексти програмного коду програмного*

*забезпечення “Карткова гра Скарбнички”*

(Вид носія даних)

(Обсяг програми (документа), арк., Кб)

*студента групи ІП-02, І курсу*

*Василенка П. О.*

Київ 2021