МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.Мечникова

Факультет математики, фізики та інформаційних технологій

Індивідуальне завдання

з дисципліни «Програмування (Об'єктно-орієнтовані мови та технології)» на тему: «СТВОРЕННЯ ІЄРАРХІЇ КЛАСІВ НА ТЕМУ "ПОРТ"»

Студентки 1 курсу, денної форми навчання. спеціальності 123 «Комґ'ютерна інженерія»

Ланевич Ольги Вікторівни

Керівник: Антоненко О.С

Одеса 2023

Зміст

[1.Постановка задачі 3](#_Toc138525816)

[2.Програмна реалізація 4](#_Toc138525817)

[**2.1 Діаграма класів** 4](#_Toc138525818)

[**2.2. Опис структури класів** 5](#_Toc138525819)

[2.2.1. Клас "Ship": 5](#_Toc138525820)

[2.2.2. Клас "Passengership": 9](#_Toc138525821)

[2.2.3. Клас "CargoShip": 11](#_Toc138525822)

[3. Робота з DOM 12](#_Toc138525823)

[**3.1 Функція для фільтрації кораблів** 12](#_Toc138525824)

[**3.2 функція addOrderedShip.** 14](#_Toc138525825)

[**3.3 Модальне вікно для створення нового корабля** 15](#_Toc138525826)

[3.3.1 Створення корабля 15](#_Toc138525827)

[3.3.2 Таблиця для кораблів 16](#_Toc138525828)

[**3.4 Модальне вікно для модифікації корабля** 17](#_Toc138525829)

[4. Демонстрація інтерфейсу 18](#_Toc138525830)

[**4.1. Створення корабля:** 18](#_Toc138525831)

[**4.2. Модифікація корабля:** 19](#_Toc138525832)

[**4.3. Видалення корабля:** 20](#_Toc138525833)

[**4.4 Перевірка та додавання шлюпок** 20](#_Toc138525834)

[5.Висновок 23](#_Toc138525835)

[6. Література 24](#_Toc138525836)

# 1.Постановка задачі

Створення ієрархії класів на тему «Порт»

Создать иерархию классов: корабль – базовый класс и пассажирский корабль – производный. Корабль имеет мощность двигателя, водоизмещение, название, порт приписки, экипаж. Пассажирский корабль имеет дополнительные поля: количество пассажиров, количество шлюпок, вместимость шлюпки. Реализовать метод проверки, что шлюпок хватает на пассажиров и членов экипажа и метод увеличения числа шлюпок до минимально необходимого, если их не хватает. Также реализовать класс грузовой корабль с дополнительным параметром – грузоподъемность.

В основном проекте реализовать создание, модификацию и удаление кораблей, а также добавление шлюпок на пассажирские корабли.

# 2.Програмна реалізація

## **2.1 Діаграма класів**

***Ship***

Name

EnginePower

Displacement

Crew

Homeport

deleteShip()

updateShip()

searchShips()

getShipDetails()

*PassengerShip*

passengerCount

lifeboatsCount

lifeboatsCapacity

checkLifeboats()

increaseLifeboats()

getShipDetails()

updateShip()

*CargoShip*

cargoCapacity

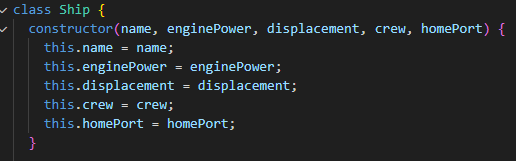
updateShip()

getShipDetails()

## **2.2. Опис структури класів**

### 2.2.1. Клас "Ship":

Клас "Ship" є базовим класом для всіх типів кораблів. Він містить загальні властивості та методи, які спільні для всіх кораблів.



Властивості класу "Ship":

*name*: представляє назву корабля.

*enginePower*: представляє потужність двигуна корабля.

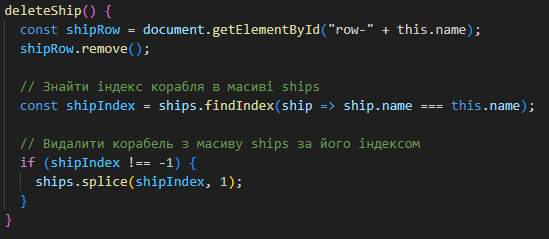
*displacement*: представляє водотоннажність корабля.

*crew*: представляє кількість екіпажу на кораблі.

*homePort:* представляє порт приписки корабля.

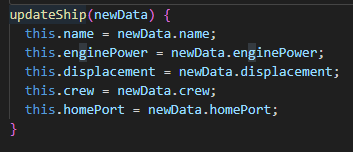
Методи класу "Ship":

Наяві 2 метода в цбому класі: *deleteShip()* та *updateShip().*



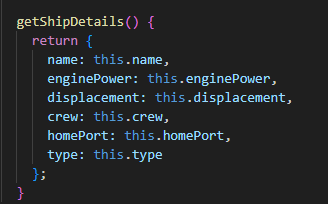
*deleteShip()*: видаляє корабель.

Цей метод видаляє корабель зі списку та з масиву ships. Спочатку він знаходить рядок, що відповідає даному кораблю у списку, за допомогою ідентифікатора row-<ім'я корабля>, та видаляє його. Далі, він знаходить індекс корабля у масиві ships за допомогою методу findIndex(), перевіряючи, чи ім'я корабля співпадає з шуканим. Якщо корабель знайдено, він видаляється з масиву ships, використовуючи метод splice()



*updateShip():* оновлює дані корабля(модифікує кораблі).

Метод updateShip(newData): Цей метод оновлює дані корабля з новими даними, переданими у вигляді об'єкту newData. Він оновлює всі основні атрибути корабля, такі як name, enginePower, displacement, crew, homePort.



getShipDetails(): Цей метод повертає об'єкт, що містить деталі про корабель. Повернуті атрибути включають name (ім'я корабля), enginePower (потужність двигуна), displacement (водотоннажність), crew (екіпаж), homePort (порт приписки) і type (тип корабля).

static searchShips(ships, shipType, isEnginePowerInRange, shipPower, isDisplacementInRange, shipDisplacement, peopleCount, PassengerShip, displaySearchResults) {

    const filteredShips = ships.filter(ship => {

      // Фільтруємо за типом корабля

      console.log(shipType);

      if (shipType !== 'all' && !isCorrectShipType(ship, shipType)) {

        return false;

      }

      function isCorrectShipType(ship, shipType) {

        switch (shipType) {

          case 'passenger':

            return ship instanceof PassengerShip;

          case 'cargo':

            return ship instanceof CargoShip;

          default:

            return true;

        }

      }

      // Фільтруємо за потужністю двигуна

      if (shipPower !== 'all' && !isEnginePowerInRange(ship.enginePower, shipPower)) {

        return false;

      }

      // Фільтруємо за водозміщенням

      if (shipDisplacement !== 'all' && !isDisplacementInRange(ship.displacement, shipDisplacement)) {

        return false;

      }

      // Фільтруємо за кількістю людей

      if (peopleCount && ship instanceof PassengerShip && ship.passengerCount < peopleCount) {

        return false;

      }

      return true;

    });

    displaySearchResults(filteredShips);

  }

searchShips(): Метод searchShips є статичним методом, який приймає наступні аргументи:

ships: масив кораблів для пошуку

shipType: тип корабля для пошуку ('all', 'passenger' або 'cargo')

isEnginePowerInRange: функція для перевірки чи потужність двигуна корабля знаходиться у заданому діапазоні

shipPower: діапазон потужності двигуна для пошуку

isDisplacementInRange: функція для перевірки чи водозміщення корабля знаходиться у заданому діапазоні

shipDisplacement: діапазон водозміщення для пошуку

peopleCount: мінімальна кількість людей на кораблі для пошуку

PassengerShip: клас пасажирського корабля

displaySearchResults: функція для виводу результатів пошуку

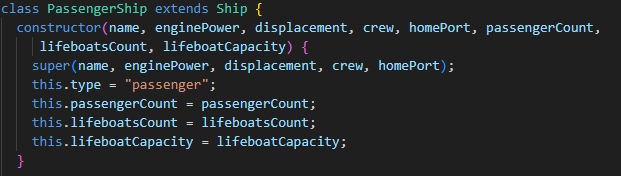
Метод searchShips фільтрує масив кораблів за заданими критеріями та викликає функцію displaySearchResults з відфільтрованим масивом кораблів.

Функція фільтрує кораблі за типом (shipType), потужністю двигуна (isEnginePowerInRange та shipPower), водозміщенням (isDisplacementInRange та shipDisplacement) та кількостю людей на кораблі (peopleCount). Якщо корабель не відповідає одному з критеріїв, то він не буде включений у результати пошуку.

По завершенню функції фільтрації кораблів метод searchShips викликає функцію displaySearchResults з відфільтрованим масивом кораблів. Ця функція використовується для виводу результатів пошуку на екран.

### 2.2.2. Клас "Passengership":

Клас "Passengership" є похідним від класу "Ship" і розширює його функціональність для пасажирських кораблів.



Додаткові властивості класу "Passengership":

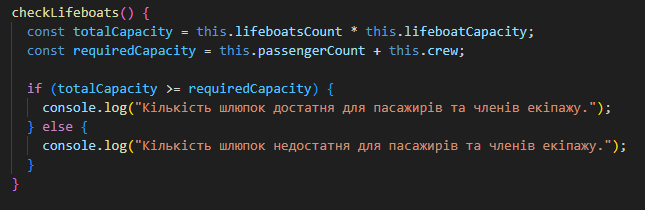
*passengerCount*: представляє кількість пасажирів, які можуть знаходитися на кораблі.

*lifeboatsCount*: представляє кількість шлюпок на кораблі.

*lifeboatCapacity*: представляє вмістимість кожної шлюпки.

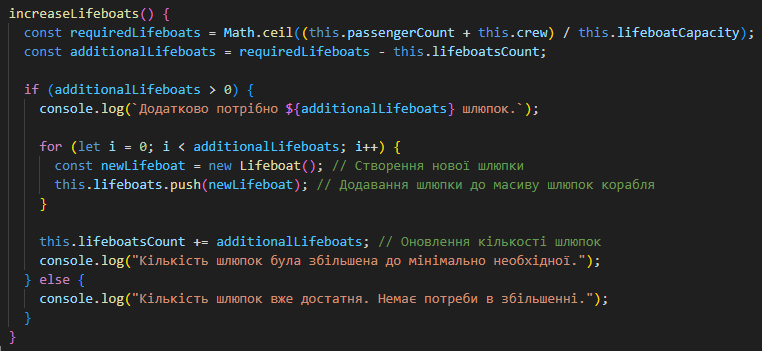
*type*: додає тип корабля зі значенням «пасажирський».

Додаткові методи класу "Passengership":



*checkLifeboats()*: перевіряє наявність шлюпок та їх готовність до використання.

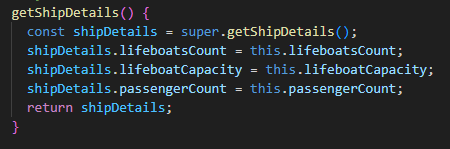
Для перевірки чи кількість шлюпок на кораблі достатня для пасажирів та членів екіпажу використовується метод checkLifeboats(passengerCount). Цей метод приймає кількість пасажирів (passengerCount) і порівнює її з максимальною ємністю шлюпок на кораблі. Якщо загальна ємність шлюпок більша або рівна необхідній ємності для пасажирів та членів екіпажу, відображається повідомлення "Кількість шлюпок достатня для пасажирів та членів екіпажу". У протилежному випадку виводиться повідомлення "Кількість шлюпок недостатня для пасажирів та членів екіпажу".



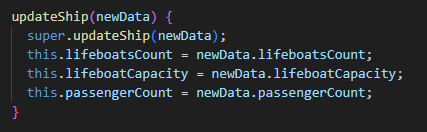
*increaseLifeboats()*: збільшує кількість шлюпок на кораблі.

Якщо під час перевірки виявляється, що кількість шлюпок недостатня для пасажирів та членів екіпажу, викликається метод increaseLifeboats(passengerCount). Цей метод обчислює необхідну кількість шлюпок, щоб забезпечити достатню ємність для всіх осіб на кораблі, і додає додаткові шлюпки до мінімально необхідної кількості. Потім виводиться повідомлення, яке показує кількість додаткових шлюпок, які були додані, і підтверджує, що кількість шлюпок була збільшена до мінімально необхідної.

Описані методи checkLifeboats(passengerCount) і increaseLifeboats(passengerCount) можна використати для перевірки наявності та додавання шлюпок на кораблі. Вони доповнюють функціональність класу PassengerShip, дозволяючи переконатись, що кількість шлюпок задовольняє вимоги щодо безпеки пасажирів та екіпажу.



*getShipDetails()*: це метод для консолі, він показує відомості про створений пасажирський корабель.Зроблено за допомогою полімарфізму.

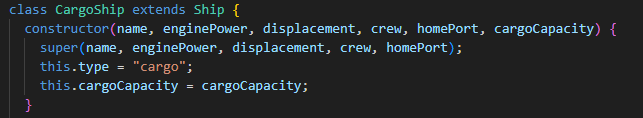


*updateShip():* оновлює дані корабля(модифікує кораблі).

Метод updateShip(newData): Цей метод оновлює дані корабля з новими даними, переданими у вигляді об'єкту newData. Метод зроблений за допомогою поліморфізму, тому оновлюються також атрибути, що відносяться до пасажирського корабля, які включають lifeboatsCount, lifeboatCapacity і passengerCount.

### 2.2.3. Клас "CargoShip":

Клас "CargoShip" є похідним від класу "Ship" і розширює його функціональність для вантажних кораблів.

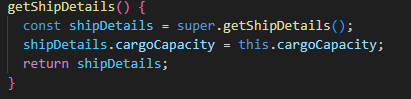


Додаткові властивості класу "CargoShip":

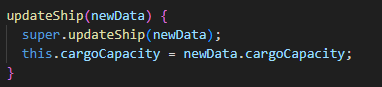
*type*: додає тип корабля зі значенням «пасажирський».

*cargoCapacity*: представляє вантажопідйомність корабля.

Додаткові методи класу " CargoShip ":



*getShipDetails()*: це метод для консолі, він показує відомості про створений грузовий корабель. Зроблено за допомогою полімарфізму.



*updateShip():* оновлює дані корабля(модифікує кораблі).

Метод updateShip(newData): Цей метод оновлює дані корабля з новими даними, переданими у вигляді об'єкту newData. Метод зроблений за допомогою поліморфізму, тому оновлюється атрибут cargoCapacity.

Вищевказані класи утворюють ієрархію, де "Passengership" і "CargoShip" успадковують властивості та методи від базового класу "Ship". Ця структура дозволяє легко розширювати функціональність і додавати нові типи кораблів, які можуть мати специфічні властивості та методи.

# 3. Робота з DOM

## **3.1 Функція для фільтрації кораблів**

Цей код додає обробник події на подію submit форми shipForm. Коли форма відправляється, виконується наступний код:

1. Запобігання стандартній поведінці форми за допомогою event.preventDefault().

shipForm.addEventListener("submit", function(event) {

    event.preventDefault();

//…

}

2. Очищення списку кораблів перед новим пошуком.

3.Отримання значень з полів форми: тип корабля (shipType), потужність двигуна (shipPower), водозміщення (shipDisplacement) та кількість людей (peopleCount).

const shipType = document.getElementById('ship-type').value;

    const shipPower = document.getElementById('ship-power').value;

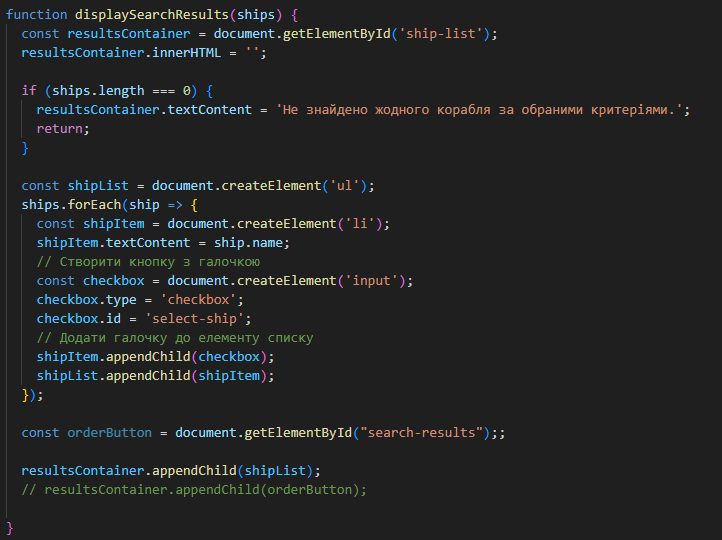
    const shipDisplacement = document.getElementById('ship-displacement').value;

    const inputElement = document.getElementById('people-count');

    const peopleCount =  Number(inputElement.value);

4.Створення функцій isEnginePowerInRange та isDisplacementInRange для перевірки чи потужність двигуна та водозміщення корабля знаходяться у заданому діапазоні.

5. Створення функції displaySearchResults для виводу результатів пошуку на екран. Ця функція очищує контейнер результатів пошуку, перевіряє чи є результати пошуку та виводить їх на екран у вигляді списку з чекбоксами.



6. Виклик статичного методу searchShips класу Ship для пошуку кораблів за заданими критеріями.

Ship.searchShips(ships, shipType, isEnginePowerInRange, shipPower, isDisplacementInRange, shipDisplacement, peopleCount, PassengerShip, displaySearchResults);

В коді також додається обробник події на натискання кнопки “Фільтрувати”. Коли кнопка натискається, виконується наступний код: знаходження всіх елементів списку з чекбоксами, перевірка чекбоксів та додавання кораблей до списку замовлених кораблів за допомогою функції addOrderedShip.

## **3.2 функція addOrderedShip.**

function addOrderedShip(shipName) {

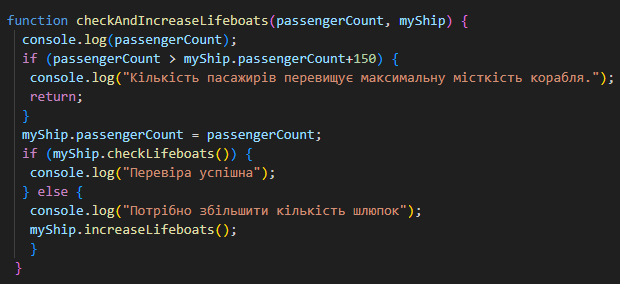
//…

}

Функція addOrderedShip приймає на вхід назву корабля (shipName) та додає його до списку замовлених кораблів. Функція перевіряє, чи вже існує елемент списку з такою назвою корабля, знаходить корабель за назвою у масиві ships, створює елемент списку для корабля та додає його до списку замовлених кораблів. Якщо корабель є пасажирським, то функція також створює кнопку “Додати пасажирів” та додає обробник події на натискання цієї кнопки.

Коли кнопка натискається, виконується наступний код: відображення модального вікна для введення кількості пасажирів, збереження кількості пасажирів у змінну passengerCount, перевірка шлюпок та збільшення їх за допомогою функції checkAndIncreaseLifeboats.

Функція checkAndIncreaseLifeboats приймає на вхід кількість пасажирів (passengerCount) та об’єкт корабля (myShip) та перевіряє шлюпки та збільшує їх за необхідності.



Перевіряються шлюпки за допомогою методу checkLifeboats об’єкта myShip. Якщо перевірка успішна, то функція повертається. Якщо перевірка не успішна, то збільшення кількості шлюпок за допомогою методу increaseLifeboats об’єкта myShip.

## **3.3 Модальне вікно для створення нового корабля**

### 3.3.1 Створення корабля

document.getElementById("buttonSubmit").addEventListener("click", function(event) {

  event.preventDefault();

//…

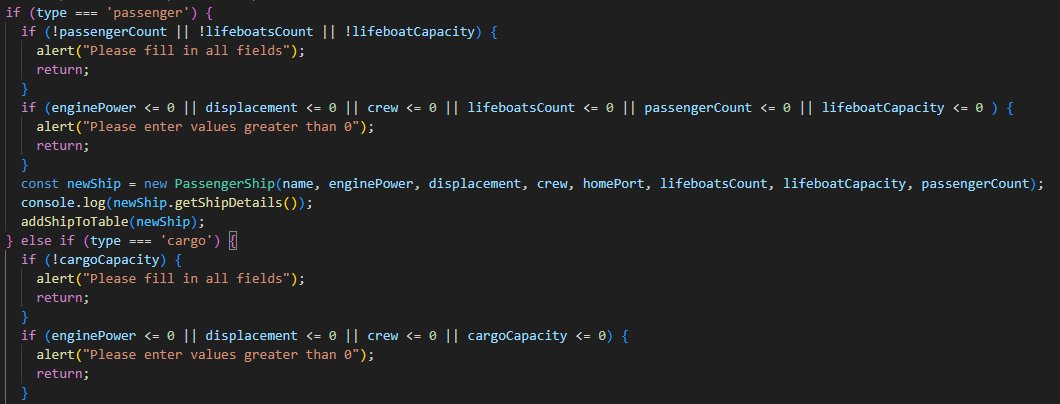
}

Цей код додає обробник події на натискання кнопки з ідентифікатором buttonSubmit. Коли кнопка натискається, виконується наступний код:

1. Запобігання стандартній поведінці форми за допомогою event.preventDefault().

2. Отримання значень з полів форми: назва корабля (name), тип корабля (type), кількість пасажирів (passengerCount), кількість шлюпок (lifeboatsCount), місткість шлюпки (lifeboatCapacity), вантажопідйомність (cargoCapacity), потужність двигуна (enginePower), водозміщення (displacement), кількість екіпажу (crew) та домашній порт (homePort).

3.Перевірка, чи всі поля форми заповнені. Якщо не всі поля заповнені, то виводиться повідомлення про помилку та функція повертається. Створення нового корабля з отриманими значеннями. Якщо тип корабля - пасажирський, то створюється новий об’єкт класу PassengerShip. Якщо тип корабля - вантажний, то створюється новий об’єкт класу CargoShip.



4. Додавання нового корабля до таблиці за допомогою функції addShipToTable.

### 3.3.2 Таблиця для кораблів

function addShipToTable(ship) {

//…

}

Функція addShipToTable(ship) приймає об'єкт ship як аргумент і додає новий рядок до таблиці shipTable.

const row = shipTable.insertRow(-1);

Створюється новий рядок за допомогою shipTable.insertRow(-1), де -1 вказує, що рядок додаватиметься в кінці таблиці.

Ідентифікатор рядка встановлюється на основі назви корабля: row.setAttribute("id", "row-" + ship.name).

const nameCell = row.insertCell(0);

  const typeCell = row.insertCell(1);

  const lifeboatsCell = row.insertCell(2);

  const lifeboatCapacityCell = row.insertCell(3);

//…

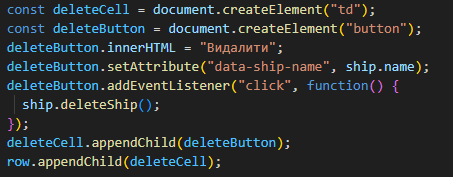
nameCell.textContent = ship.name;

  typeCell.textContent = ship.type;

  lifeboatsCell.textContent = ship.lifeboatsCount;

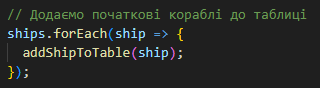
**//…**

Створюються комірки для кожного поля корабля, вставляються в рядок і заповнюються значеннями з об'єкта ship.



Додається комірка з кнопкою "Видалити". Створюється новий елемент <td> і кнопка <button>, які прив'язані до об'єкта ship. При натисканні на кнопку викликається метод deleteShip(), що видаляє корабель. Кнопка додається до комірки і комірка до рядка.

Додається комірка з кнопкою "Модифікувати". Аналогічно до кнопки "Видалити", створюється новий елемент <td> і кнопка <button>, які прив'язані до об'єкта ship. При натисканні на кнопку викликається функція openEditModal(ship), яка відкриває модальне вікно для редагування корабля. Кнопка додається до комірки і комірка до рядка.



Цей фрагмент коду додає початкові кораблі до таблиці, використовуючи функцію addShipToTable(ship) для кожного елемента у масиві ships.

5. Скидання значень полів форми.



6.Закриття модального вікна.

Цей код дозволяє користувачеві створювати новий корабель та додавати його до таблиці.

## **3.4 Модальне вікно для модифікації корабля**

function openEditModal(ship) {

//…

}

Функція openEditModal(ship) відкриває модальне вікно для редагування корабля з переданим об'єктом ship.

 Зберігається посилання на поточний корабель у змінній currentShip.

Значення полів модального вікна заповнюються властивостями поточного корабля, використовуючи document.getElementById(), також отримуються посилання на додаткові поля для пасажирського корабля (passengerFields) і вантажного корабля (cargoFields), а потім ці поля приховуються, встановлюючи їх стиль display на "none".

Залежно від типу корабля (PassengerShip або CargoShip), відповідні додаткові поля відображаються, встановлюючи їх стиль display на "block". Значення цих полів також заповнюються властивостями поточного корабля.

document.getElementById("myButton").addEventListener("click", function(event) {

    event.preventDefault();

/…

}

Додається обробник події click для кнопки "Зберегти" (myButton). При натисканні на кнопку викликається функція, яка перевіряє значення полів, отримує нові дані з полів модального вікна і оновлює значення відповідних комірок таблиці.

const shipRow = document.getElementById("row-" + currentShip.name);

Оновлюється ідентифікатор рядка таблиці на основі нової назви корабля.

За допомогою shipRow.cells[i].innerHTML оновлюються значення відповідних комірок таблиці.

Викликається метод updateShip(newData) корабля currentShip, щоб оновити його властивості з новими даними.

# 4. Демонстрація інтерфейсу

## **4.1. Створення корабля:**

Натисніть на кнопку "Створити корабель".

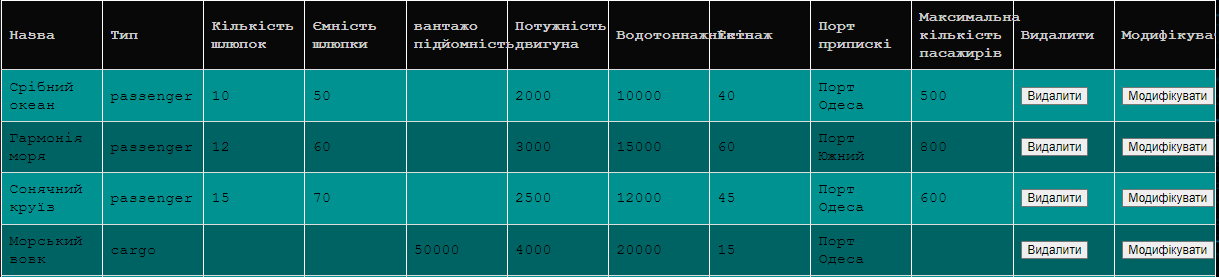
Відкриється вікно створення корабля, де ви можете вибрати різні параметри для вашого корабля, наприклад, назву, тип, розмір і особливості.



Після встановлення бажаних параметрів натисніть кнопку "Створити"



Програма створить новий корабель з вказаними вами характеристиками



та додасть до таблиці нижче.

## **4.2. Модифікація корабля:**

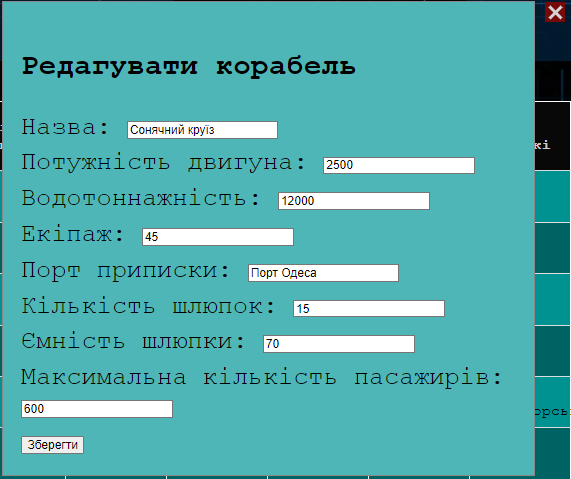
Виберіть корабель, який ви хочете модифікувати, зі списку на екрані.

Натисніть кнопку "Модифікувати" 

З'явиться вікно редагування, де ви зможете змінити параметри корабля, наприклад, тип, назва чи екіпаж.

Якщо це пасажирський корабель то з’являться додаткові поля «кількість шлюпок», «ємність шлюпки» та «максимальна кількість пасажирів».

Якщо це грузовий корабель то з’явиться додаткове поле «вантажопідйомність».



Вище показано вікно для модифікації пасажирського корабля.

Після внесення необхідних змін натисніть кнопку "Зберегти", щоб застосувати модифікації до корабля. 

## **4.3. Видалення корабля:**

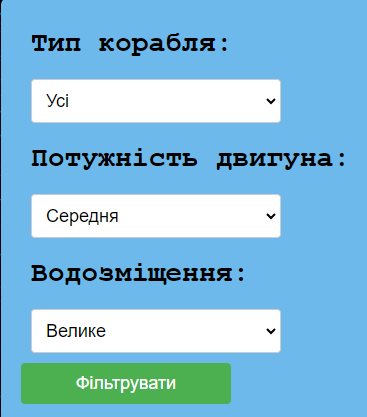
Знайдіть корабель, який ви хочете видалити, у списку на екрані.

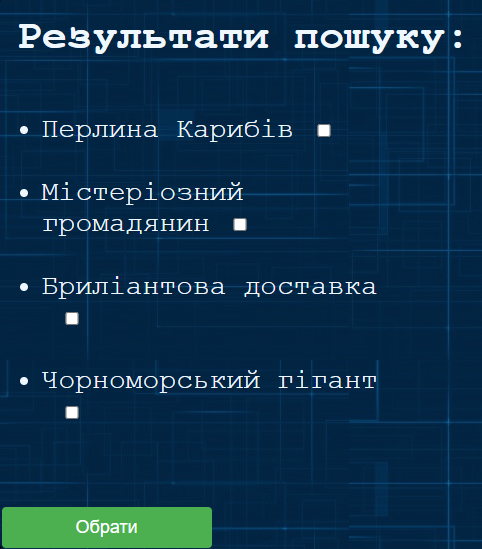
Виберіть корабель, натиснувши на кнопку видалити 

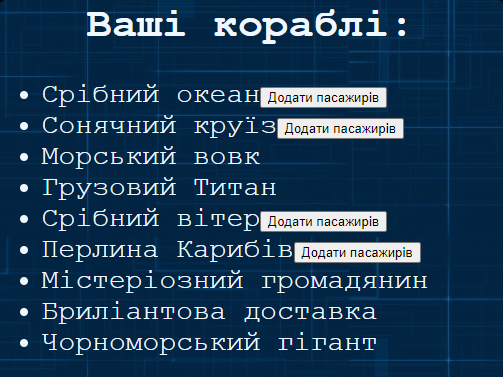
Підтвердьте видалення корабля, якщо буде потрібно, і корабель буде вилучений з програми.

## **4.4 Перевірка та додавання шлюпок**

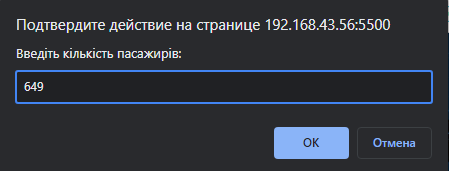
Цей метод використовується в частині із замовленням корабля

Спочатку обираємо ті параметри за якими будемо фільтрувати корабель

 Ставимо галочкі навпроти біля того корабля який хочемо обрати та натискаємо кнопку Обрати



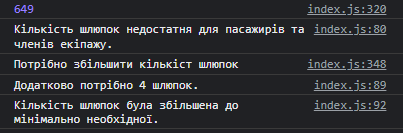
Навпроти пасажирського корабля з’явиться кнопка Додати пасажирів



Вводимо потрібну кількість пасажирів та натискаємо на ОК.

Далі користувачу нічого не показується і автоматично перевіряється чи шлюпок вистачає на всіх пасажирів.

Для того щоб відслідкувати цей процес, то потрібно відкрити інструменти для розробника, й подивитись як виконується перевірка



# 5.Висновок

У ході демонстрації інтерфейсу програми "Порт Одеси" були представлені основні кроки, необхідні для створення корабля, його модифікації та видалення. Починаючи з запуску програми, користувач може створити свій власний корабель, вказавши різні параметри, такі як назва, тип, розмір і особливості. Потім програма надає можливість модифікувати корабель, дозволяючи змінювати характеристики. Крім того, користувач може видалити непотрібний корабель за допомогою відповідної опції.

checkLifeboats(passengerCount) і increaseLifeboats(passengerCount) дозволяють забезпечити безпеку пасажирів та членів екіпажу на пасажирських кораблях.

В результаті, програма "Порт Одеси" надає зручний і простий у використанні інтерфейс, який дозволяє користувачам створювати, модифікувати та видаляти кораблі в портовій симуляції. Цей інтерфейс може бути корисним для експериментування з різними конфігураціями кораблів та дослідженням їх характеристик у віртуальному порту Одеси.

У ході демонстрації інтерфейсу програми "Порт Одеси" було показано не тільки зручний і простий у використанні інтерфейс, але й деталі коду, які використовуються для реалізації функціональності програми. Програма "Порт Одеси" використовує об'єктно-орієнтований підхід, де функціонал поділений на класи та методи.

Один з основних класів програми - "Корабель". В цьому класі описані характеристики корабля, такі як назва, тип, потужність двигуна, порт приписки та екіпаж. Клас "Корабель" має методи для модифікації та видалення корабля.

Додатково, програма включає два похідних класи: "ПасажирськийКорабель" і "ГрузовийКорабель". Ці класи успадковують характеристики базового класу "Корабель" та розширюють його функціональність. Наприклад, клас "ПасажирськийКорабель" має додаткові властивості такі як кількість шлюпок, об’єм шлюпки та кількість пасажирів, а клас "ГрузовийКорабель" – додаткове поле вантажопідйомність.

# 6. Література

1. "JavaScript: The Definitive Guide" автора Девіда Фланагана

2. MDN Web Docs - https://developer.mozilla.org/uk/docs/Web/JavaScript