МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київська державна академія водного транспорту
ім. Петра Конашевича-Сагайдачного
Факультет економіки транспорту
Кафедра програмної інженерії

КУРСОВА РОБОТА
З дисципліни «Основи програмування»

Завдання на роботу: розробити гру «Морський бій», що для введення-виведення інформації буде використовувати консоль.

Керівник роботи :
Ратушняк Т.В.

Допущений до захисту
«___»_____2016р.

Захищена з оцінкою
_____

_____
(підпис)

Виконавець:
Студент І-го курсу
Групи ПІ-1519
Калініченко М. В.

_____
(підпис)

Київ 2016

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київська державна академія водного транспорту
ім. Петра Конашевича-Сагайдачного
Факультет економіки транспорту
Кафедра програмної інженерії

Дисципліна: «Основи програмування»
Спеціальність: «Програмна інженерія»

Курс: «І» Група: ПІ-1519 Семестр: ІІ

Завдання
на курсову роботу студента
Калініченко Микити Віталійовича

1) **Тема роботи**: розробити гру «Морський бій», що для введення-виведення інформації буде використовувати консоль.

2) **Строк здачі студентом закінченої роботи:_____**

3) **Вихідні дані роботи:**

3.1) **На етапі встановлення кораблів:** Поточний стан поля користувача з відміченими комірками, у які не можна встановлювати новий корабель. Перелік кораблів та їх кількість, які необхідно встановити щоб почати гру.

3.2) **На етапі гри:** Відображення поточного стану поля супротивника та користувача.

Зміст пояснювальної записки(основні питання): використання взаємодії об'єктів, створення програми з консольним інтерфейсом користувача.

4) **Дата видачі завдання:_____**

Студент_____
(підпис)

Керівник_____
(підпис)

«___»_____2016 року

Зміст

Вступ

Завдання: розробити комп'ютерну гру «Морський бій». Забезпечити введення та виведення даних через консоль.

Послідовність виконання завдання

1. Розробити загальну структуру програми.
2. Спроектувати функціональну структуру програми у вигляді функціональної блок-схеми.
3. Створити клас-структуру, що зберігає дані про поточний стан полів користувача та супротивника.
4. Створити класи для обробки даних:
   4.1. Спроектувати клас, що відповідає за логіку супротивника (комп'ютера).
   4.2. Спроектувати клас, що відповідає за введення-виведення даних, а також забезпечує коректність введених користувачем даних.
   4.3. Створити клас, що буде керувати загальною логікою програми.
5. Зібрати всі класи у єдину програму, зібрати та налагодити її.
6. Розробити пояснювальну записку.

Розділ 1

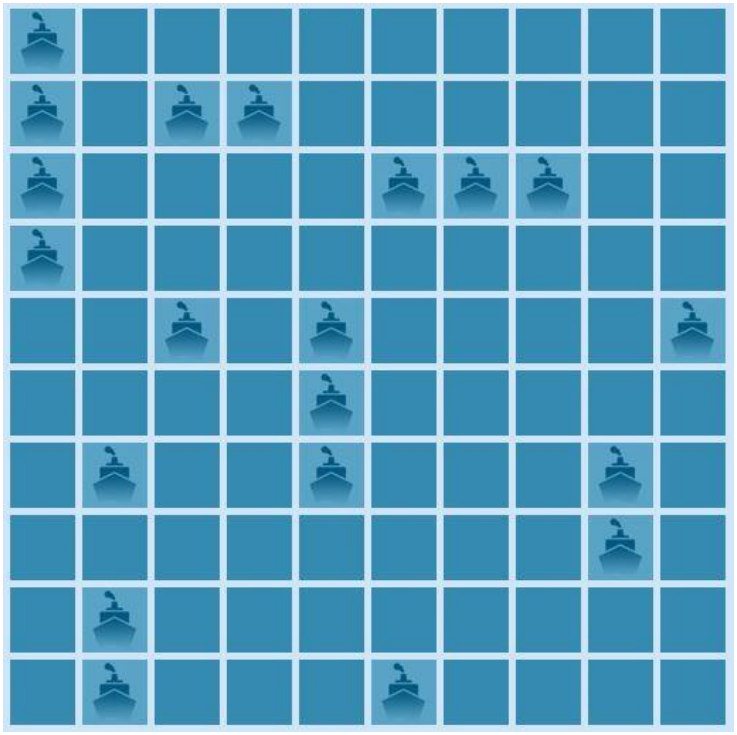
Постановка завдання, аналіз предметної області, формування вимог до розроблюваного ПЗ

Постановка завдання

Розробити комп’ютерну гру «Морський бій». Забезпечити введення та виведення даних через консоль.

Аналіз предметної області

Гра «морський бій» - це гра для двох. У ході гри, гравці спочатку креслять на папері поле 10х10, далі розставляють на цьому полі: 1 лінкор (4 підряд клітинки по горизонталі або по вертикалі), 2 крейсери (3 підряд клітинки по горизонталі або по вертикалі), 3 есмінця (2 підряд клітинки по горизонталі або по вертикалі) та 4 подвійних човна (1 клітинка). Причому, між сусідніми кораблями повинна обов’язково бути хоча б 1 порожня клітинка (див. мал. 1).



Малюнок 1. Приклад розташування кораблів.

Після встановлення кораблів, гравці почергово називають координати комірок. Після цього:

- Якщо «постріл» влучив у комірку, котру не займає жоден з кораблів супротивника, то супротивник відповідає «Мимо». Після цього, в цю комірку перший гравець більше не стріляє.

- Якщо у названій комірці супротивника розташовано багатопалубний корабель, то супротивник відповідає «Влучив» або «Поранив». Перший гравець ставить у цій комірці хрестик та отримує додатковий постріл (як правило, постріли продовжуються неподалік від цієї комірки, доти, доки не виконається умова наступного пункту).
- Якщо у названій комірці супротивника встановлено подвійний човен або знаходиться остання неуражена комірка багатопалубного корабля, то супротивник повинен сказати «Потоплено» або «Вбито». Обидва гравці позначають потоплений корабель, а перший гравець отримує додатковий постріл

Вимоги

Програма повинна робити наступне

1. Запитувати команд користувача через консоль.
2. Перевіряти синтаксис команд та забезпечувати їх коректність.
3. Відображати поле(-я) користувача та супротивника.
4. Імітувати логіку реального гравця під час гри.
5. Відображати інші необхідні елементи.

Розділ 2

Проектні рішення

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київська державна академія водного транспорту
ім. Петра Конашевича-Сагайдачного

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ «МОРСЬКИЙ БІЙ»

Ескізний проект
Пояснювальна записка

Розробив:
Студент групи: ПІ-1519
Калініченко Микита Віталійович

Київ 2016

Пояснювальна записка до ескізного проекту

Загальні положення

Даний документ є ескізним проектом на створення програмного модуля «Морський бій» (надалі - ПЗ), для гри у Морський бій з комп’ютером.

Основні технічні рішення

Рішення про структуру ПЗ

ПЗ «Морський бій» буде реалізовано згідно шаблону (патерну) проектування MVP (Model-View-Presenter, у перекладі – Модель-Представлення-Пред’явник). Цей шаблон буде використано для ізоляції між собою: візуальних компонентів, поведінки та обробки даних. Також це дозволить зробити зручнішим процес супроводу та налагодження ПЗ.

ПЗ буде складатись з таких компонентів: консольний інтерфейс користувача, загальна логіка програми, та компоненти обробки даних (компонент «мапа» і «логіка супротивника»). Кожен з компонентів буде реалізовувати свій API-інтерфейс для інкапсуляції внутрішніх методів, структур та даних, які непотрібні ззовні нього, а також для абстрагування від внутрішньої логіки компонента, під час його використання у програмуванні іншого компонента. Таким чином, для передачі даних між компонентами будуть використовуватись тільки ті методи, що зазначені в їх інтерфейсах

Інтерфейс користувача (у даному патерні проектування - Представлення) буде запитувати дані (команди з відповідними параметрами) у користувача, виводити певні повідомлення у консоль та перевіряти правильність синтаксису введених команд і їх логічну коректність.

Загальна логіка ПЗ (у даному патерні проектування – Пред’явник) буде керувати основними процесами що відбуваються на поточний момент. А саме – запитувати у Представлення команди, обробляти їх самостійно або надсилати на обробку до Моделі, в залежності від команди. Керувати ходами користувача та логіки супротивника згідно правил гри. Запитувати у Моделі хто виграв та виводити відповідне повідомлення.

Мапа (Модель) буде обробляти структуру даних, у вигляді якої представлені поля гравця та супротивника. Приводити, у разі запиту Пред’явника, цю(ці) структуру(-и) даних у формат, що сприймає Представлення. Перевіряти чи не виграв гравець або комп’ютер. Обробляти ходи гравця.

Логіка супротивника (Модель) буде генерувати поле супротивника та робити постріли «імітуючи» поведінку реальної людини.

Опис взаємозв’язків Пред’явника з Моделями та Представленням а також опис загального алгоритму ПЗ наведено у Додатку 1 до ескізного проекту. Опис структур та API-інтерфейсів наведено у Додатку 2 до ескізного проекту.

Рішення про режими функціонування ПЗ

ПЗ буде працювати в однокористувацькому режимі з використанням псевдо-людської логіки гри у Морський бій.

Склад функцій та комплексів задач що реалізує ПЗ

ПЗ повинно реалізовувати наступні функції:

1. Запитувати дані та команди у користувача.
2. Перевіряти синтаксис та аналізувати достовірність параметрів команд, що ввів користувач.
3. Підтримка ігрової логіки.
4. Імітування логіки супротивника.

Рішення про склад програмних засобів, мов, алгоритмів і процедур, а також методи їх реалізації

Для реалізації ПЗ буде використано IDE NetBeans, IDE IntelliJ IDEA, Sublime Text 3, ДСКВ Git, мова Java та остання версія Java JDK – 8.1.

Загальний алгоритм ПЗ та API-функції його компонентів зазначені у Додатку 1 та Додатку 2 до даного документу відповідно.

Джерела розробки

Даний документ розроблявся на підставі ДСТУ 19.404 – 79.

Додатки до ескізного проекту

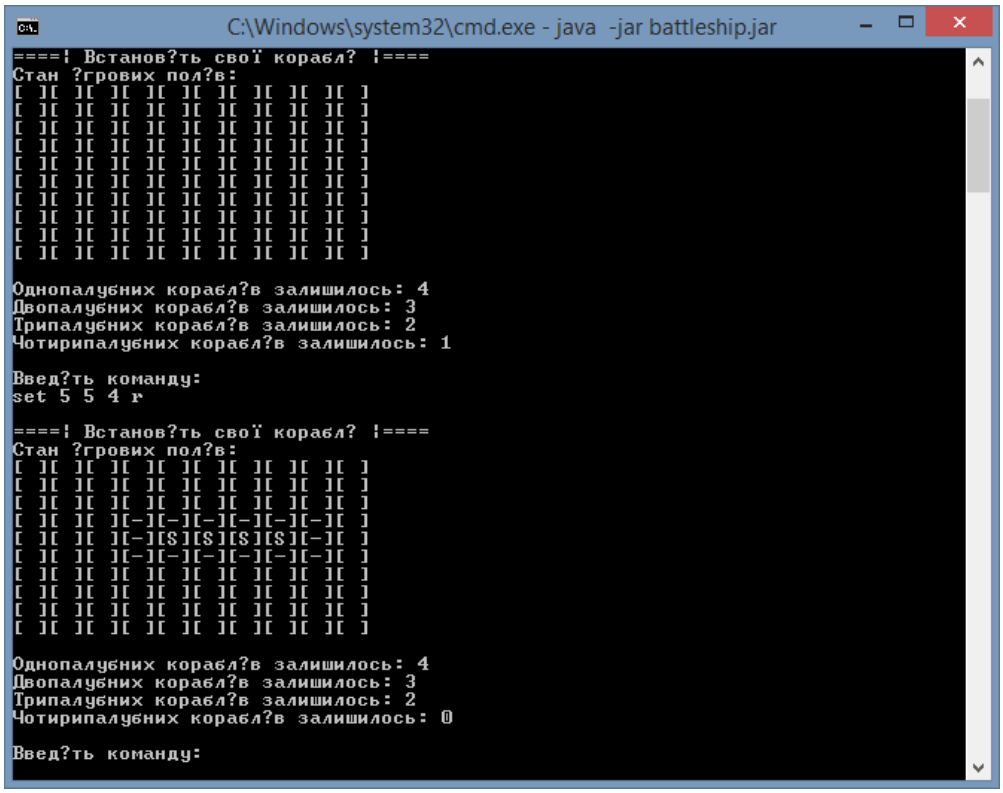
ТУТ ПОТІМ БУДЕ ОКРЕМИЙ ЛИСТ ЗІ СХЕМОЮ

ТУТ ПОТІМ БУДЕ ОКРЕМИЙ ЛИСТ ЗІ СХЕМОЮ

ТУТ ПОТІМ БУДЕ ОКРЕМИЙ ЛИСТ ЗІ СХЕМОЮ

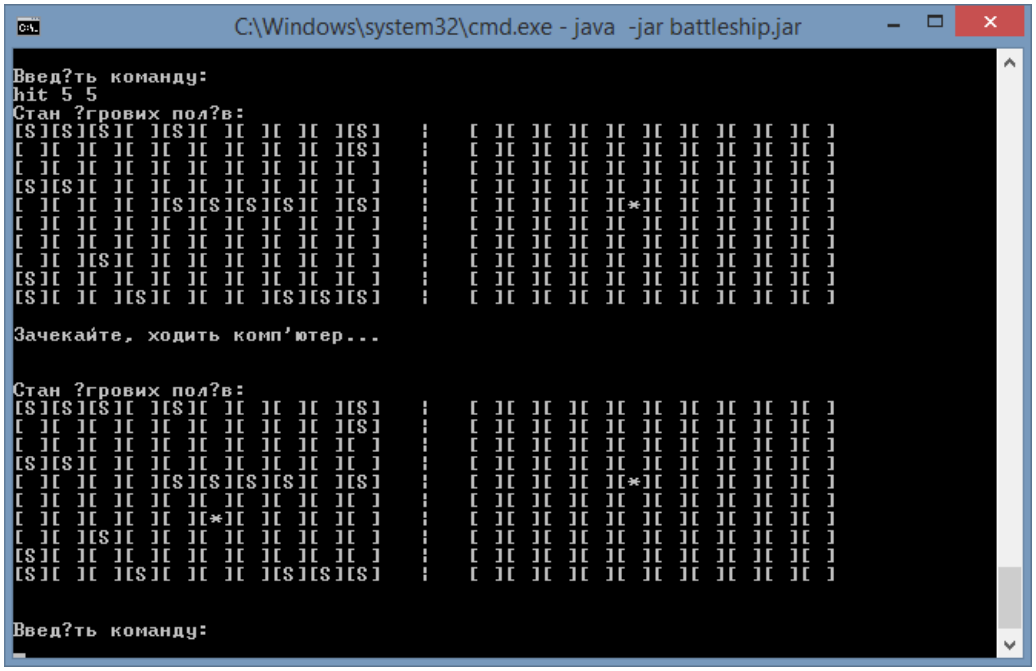
ДОДАТКИ ДО КУРСОВОЇ
Додаток 1. Скріншоти програми

Режим налаштування мапи користувача



Нажаль, поточні налаштування операційної системи не дозволяють використання букви «і», тому консоль замінила її на «?».

Режим гри



Додаток 2. Програмний код

Вихідний текст API-інтерфейсу IUser:

```
package battleship;

public interface IUser
{
   void ShowMessage(String message);
   void DisplayMap(String map);

   String[] AskCommand();
}
```

Вихідний текст API-інтерфейсу IControlLogic:

```
package battleship;

public interface IControlLogic
{
   void StartTheGame();
}
```

Вихідний текст API-інтерфейсу IMap:

```
package battleship;
/**
 * This interface defines actions with map to use from class, which implements IContiolLogic
 * @author Nk185
 */
public interface IMap
{
   String GetMaps(boolean showEnemyShips);
   String GetSpecifiedMap(MapSettings ms);

   MapSettings GetUserMap();

   boolean HitEnemyCell(int x, int y);
   boolean CheckVictory(boolean checkPlayerMap);
   boolean SetUserMap(MapSettings userMap);
   boolean SetEnemyMap(MapSettings enemyMap);
}
```

Вихідний текст API-інтерфейсу IEnemyLogic:

```
package battleship;

public interface IEnemyLogic
{
   void Fire(MapSettings userMap);
   MapSettings GenerateEnemyMap();
}
```

Вихідний текст перерахування ECellStatus:

```
package battleship;
/**
 * An enumeration of all possible cells conditions.
 * @author Nk185
 */
public enum ECellStatus
{
   /**
```

```
    * Empty opened cell.
    */
   Empty,
   /**
    * Cell contains whole ship or it's part.
    */
   ContainsShip,
   /**
    * Near this cell ship is located.
    * Note: reserved only for map setup process. In game assumed like ClosedEmpty.
    */
   LocatedNearShip,
   /**
    * This cell is not opened by user yet, but it's empty.
    */
   ClosedEmpty,
   /**
    * This cell contains whole ship or it's part and it was hited by user.
    */
   Hited;
}
```

Вихідний текст класу UserInterface:

```
package battleship;

import java.util.Scanner;

public class UserInterface implements IUser
{

   public UserInterface()
   {

   }

   /**
    * Displays user's and enemy's maps
    * @param map two maps in string format;
    */
   @Override
   public void DisplayMap(String map)
   {
      System.out.println("Стан ігрових полів:\n" + map);
   }
   /**
    * Displays specified message.
    * @param message message to display;
    */
   @Override
   public void ShowMessage(String message)
   {
      System.out.println(message);
   }


   /**
    * Asks user for a command.
    * <p>Possible commands:
    * <ul>
    *   <li>"set x y t dr", where t - is a type of ship [1..4] and dr is a direction [left (l), right(r), up(u), down(d)];</li>
    *   <li>"hit x y"</li>
```

```
 *   <li>"reset"</li>
 * </ul>
 * @return verified array of strings with command.
 * <p><i>Example: {"set","2","7","4", "u"}</i>;
 */
@Override
public String[] AskCommand()
{
   boolean success = false;
   String command;
   String[] resCommand;

   do
   {
      do
      {
         System.out.println("\nВведіть команду:");
         command = new Scanner(System.in).nextLine();
      }
      while ((!command.contains("set")) && (!command.contains("hit")) && (!command.contains("reset")));


      resCommand = command.split(" ");

      switch(resCommand[0])
      {
         case "set":
            {
               if (resCommand.length == 5)
                  success = isValidDigit(resCommand[1]) && isValidDigit(resCommand[2]) &&
                     isValidShipType(resCommand[3]) && isValidDirec(resCommand[4]);
               else
                  success = false;

               break;
            }
         case "hit":
            {
               if (resCommand.length == 3)
                  success = isValidDigit(resCommand[1]) && isValidDigit(resCommand[2]);
               else
                  success = false;

               break;
            }
         case "reset":
            {
               success = true;
               break;
            }
      }

      if (!success)
         System.out.println("Невірний формат команди. Спробуйте знову.");

   } while (!success);

   return resCommand;
}


/**
```

```
    * Verifies if input string possible to convert to integer.
    * And also if parameter in map range.
    * @param input string line to verify
    * @return true if acceptable, false if not.
    */
   private boolean isValidDigit(String input)
   {
      int i;
      boolean result;

      try
      {
         i = Integer.parseInt(input);

         result = (i >= 1) && (i <= 10);
      }
      catch (Exception e) { result = false; }

      return result;
   }
   /**
    * Verifies if input string possible to convert to integer.
    * And also if parameter represent ship length.
    * @param input string line to verify
    * @return true if acceptable, false if not.
    */
   private boolean isValidShipType(String input)
   {
      int i;
      boolean result;

      try
      {
         i = Integer.parseInt(input);

         result = (i >= 1) && (i <= 4);
      }
      catch (Exception e) { result = false; }

      return result;
   }
   /**
    * Verifies if input string possible to convert to direction.
    * And also if parameter represent ship direction.
    * @param input string line to verify
    * @return true if acceptable, false if not.
    */
   private boolean isValidDirec(String input)
   {
      return ("l".equals(input)) || ("r".equals(input)) || ("u".equals(input)) || ("d".equals(input));
   }
}
```

Вихідний текст класу MapEngine:

```
package battleship;

class MapEngine
{

   // MapSettings (distMap) ALWAYS WILL BE PASSED BY REFERENCE!!!!!
   public static final boolean SetShipByCoord(MapSettings distMap, int xPos, int yPos, int boardCount, String
direction)
```

```
{
   boolean status = false;
   boolean isSetUpAllowed = true;

   switch (direction)
   {
      case "l":
         if (xPos - boardCount >= 0)
            status = true;
         break;
      case "r":
         if (xPos + boardCount <= 11)
            status = true;
         break;
      case "u":
         if (yPos - boardCount >= 0)
            status = true;
         break;
      case "d":
         if (yPos + boardCount <= 11)
            status = true;
         break;
   }

   xPos--;
   yPos--;

   if (status)
   {
      switch (direction)
      {
         case "l":
         {
            for (int i = xPos; i > xPos - boardCount; i--)
            {
               if ((distMap.Map[i][yPos] == ECellStatus.ContainsShip)
                     || (distMap.Map[i][yPos] == ECellStatus.LocatedNearShip))
               {
                  isSetUpAllowed = false;
                  status = false;
               }
            }

            if (isSetUpAllowed)
            {
               for (int i = xPos; i > xPos - boardCount; i--)
               {
                  distMap.Map[i][yPos] = ECellStatus.ContainsShip;
                  if (yPos + 1 <= 9)
                     distMap.Map[i][yPos + 1] = ECellStatus.LocatedNearShip;
                  if (yPos - 1 >= 0)
                     distMap.Map[i][yPos - 1] = ECellStatus.LocatedNearShip;

                  if (xPos + 1 <= 9)
                     for (int j = yPos - 1; j <= yPos + 1; j++)
                     {
                        if (j <= 9 && j >= 0)
                           distMap.Map[xPos + 1][j] = ECellStatus.LocatedNearShip;
                     }

                  if (xPos - boardCount >= 0)
                     for (int j = yPos - 1; j <= yPos + 1; j++)
```

```
            {
                if (j <= 9 && j >= 0)
                    distMap.Map[xPos - boardCount][j] = ECellStatus.LocatedNearShip;
            }
        }
    }

    break;
}
case "r":
{
    for (int i = xPos; i < xPos + boardCount; i++)
    {
        if ((distMap.Map[i][yPos] == ECellStatus.ContainsShip)
            || (distMap.Map[i][yPos] == ECellStatus.LocatedNearShip))
        {
            isSetUpAllowed = false;
            status = false;
        }
    }

    if (isSetUpAllowed)
    {
        for (int i = xPos; i < xPos + boardCount; i++)
        {
            distMap.Map[i][yPos] = ECellStatus.ContainsShip;
            if (yPos + 1 <= 9)
                distMap.Map[i][yPos + 1] = ECellStatus.LocatedNearShip;
            if (yPos - 1 >= 0)
                distMap.Map[i][yPos - 1] = ECellStatus.LocatedNearShip;

            if (xPos + boardCount <= 9)
                for (int j = yPos - 1; j <= yPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[xPos + boardCount][j] = ECellStatus.LocatedNearShip;
                }

            if (xPos - 1 >= 0)
                for (int j = yPos - 1; j <= yPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[xPos - 1][j] = ECellStatus.LocatedNearShip;
                }
        }
    }

    break;
}
case "u":
{
    for (int i = yPos; i > yPos - boardCount; i--)
    {
        if ((distMap.Map[xPos][i] == ECellStatus.ContainsShip)
            || (distMap.Map[xPos][i] == ECellStatus.LocatedNearShip))
        {
            isSetUpAllowed = false;
            status = false;
        }
    }

    if (isSetUpAllowed)
```

```
    {
        for (int i = yPos; i > yPos - boardCount; i--)
        {
            distMap.Map[xPos][i] = ECellStatus.ContainsShip;
            if (xPos - 1 >= 0)
                distMap.Map[xPos - 1][i] = ECellStatus.LocatedNearShip;
            if (xPos + 1 <= 9)
                distMap.Map[xPos + 1][i] = ECellStatus.LocatedNearShip;

            if (yPos + 1 <= 9)
                for (int j = xPos - 1; j <= xPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[j][yPos + 1] = ECellStatus.LocatedNearShip;
                }

            if (yPos - boardCount >= 0)
                for (int j = xPos - 1; j <= xPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[j][yPos - boardCount] = ECellStatus.LocatedNearShip;
                }
        }
    }

    break;
}
case "d":
{
    for (int i = yPos; i < yPos + boardCount; i++)
    {
        if ((distMap.Map[xPos][i] == ECellStatus.ContainsShip)
                || (distMap.Map[xPos][i] == ECellStatus.LocatedNearShip))
        {
            isSetUpAllowed = false;
            status = false;
        }
    }

    if (isSetUpAllowed)
    {
        for (int i = yPos; i < yPos + boardCount; i++)
        {
            distMap.Map[xPos][i] = ECellStatus.ContainsShip;
            if (xPos - 1 >= 0)
                distMap.Map[xPos - 1][i] = ECellStatus.LocatedNearShip;
            if (xPos + 1 <= 9)
                distMap.Map[xPos + 1][i] = ECellStatus.LocatedNearShip;

            if (yPos - 1 >= 0)
                for (int j = xPos - 1; j <= xPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[j][yPos - 1] = ECellStatus.LocatedNearShip;
                }

            if (yPos + boardCount <= 9)
                for (int j = xPos - 1; j <= xPos + 1; j++)
                {
                    if (j <= 9 && j >= 0)
                        distMap.Map[j][yPos + boardCount] = ECellStatus.LocatedNearShip;
                }
```

```
                }
            }

            break;
        }
    }
}

return status;
}

// fix vertical surrounding
public static final void SurroundShipWithEmptyCell(int xCoord, int yCoord, MapSettings distMap, char
shipDirection)
{
    if (shipDirection == 'h') // horizontalShip
    {
        int lastBoardLeft = -1;
        int lastBoardRight = -1;

        for (int i = xCoord; i <= xCoord + 5; i++)
        {
            if (i <= 9)
            {
                if (distMap.Map[i][yCoord] == ECellStatus.Hited)
                {
                    if (yCoord - 1 >= 0)
                        distMap.Map[i][yCoord - 1] = ECellStatus.Empty;
                    if (yCoord + 1 <= 9)
                        distMap.Map[i][yCoord + 1] = ECellStatus.Empty;
                } else
                {
                    lastBoardRight = i;
                    break;
                }
            } else
            {
                lastBoardRight = -1;
                break;
            }
        }

        for (int i = xCoord; i >= xCoord - 5; i--)
        {
            if (i >= 0)
            {
                if (distMap.Map[i][yCoord] == ECellStatus.Hited)
                {
                    if (yCoord - 1 >= 0)
                        distMap.Map[i][yCoord - 1] = ECellStatus.Empty;
                    if (yCoord + 1 <= 9)
                        distMap.Map[i][yCoord + 1] = ECellStatus.Empty;
                } else
                {
                    lastBoardLeft = i;
                    break;
                }
            } else
            {
                lastBoardLeft = -1;
                break;
            }
```

```
    }

    if (lastBoardLeft != -1)
    {
        distMap.Map[lastBoardLeft][yCoord] = ECellStatus.Empty;

        if (yCoord - 1 >= 0)
            distMap.Map[lastBoardLeft][yCoord - 1] = ECellStatus.Empty;

        if (yCoord + 1 <= 9)
            distMap.Map[lastBoardLeft][yCoord + 1] = ECellStatus.Empty;
    }

    if (lastBoardRight != -1)
    {
        distMap.Map[lastBoardRight][yCoord] = ECellStatus.Empty;

        if (yCoord - 1 >= 0)
            distMap.Map[lastBoardRight][yCoord - 1] = ECellStatus.Empty;

        if (yCoord + 1 <= 9)
            distMap.Map[lastBoardRight][yCoord + 1] = ECellStatus.Empty;
    }

} else if (shipDirection == 'v')
{
    int lastBoardTop = -1;
    int lastBoardBottom = -1;

    for (int i = yCoord; i <= yCoord + 5; i++)
    {
        if (i <= 9)
        {
            if (distMap.Map[xCoord][i] == ECellStatus.Hited)
            {
                if (xCoord - 1 >= 0)
                    distMap.Map[xCoord - 1][i] = ECellStatus.Empty;
                if (xCoord + 1 <= 9)
                    distMap.Map[xCoord + 1][i] = ECellStatus.Empty;
            } else
            {
                lastBoardBottom = i;
                break;
            }
        } else
        {
            lastBoardBottom = -1;
            break;
        }
    }

    if (lastBoardBottom != -1)
    {
        distMap.Map[xCoord][lastBoardBottom] = ECellStatus.Empty;

        if (xCoord - 1 >= 0)
            distMap.Map[xCoord - 1][lastBoardBottom] = ECellStatus.Empty;

        if (xCoord + 1 <= 9)
            distMap.Map[xCoord + 1][lastBoardBottom] = ECellStatus.Empty;
    }
```

```
            for (int i = yCoord; i >= yCoord - 5; i--)
            {
               if (i >= 0)
               {
                  if (distMap.Map[xCoord][i] == ECellStatus.Hited)
                  {
                     if (xCoord - 1 >= 0)
                        distMap.Map[xCoord - 1][i] = ECellStatus.Empty;
                     if (xCoord + 1 <= 9)
                        distMap.Map[xCoord + 1][i] = ECellStatus.Empty;
                  } else
                  {
                     lastBoardTop = i;
                     break;
                  }
               } else
               {
                  lastBoardTop = -1;
                  break;
               }
            }

            if (lastBoardTop != -1)
            {
               distMap.Map[xCoord][lastBoardTop] = ECellStatus.Empty;

               if (xCoord - 1 >= 0)
                  distMap.Map[xCoord - 1][lastBoardTop] = ECellStatus.Empty;

               if (xCoord + 1 <= 9)
                  distMap.Map[xCoord + 1][lastBoardTop] = ECellStatus.Empty;
            }
         }
      }

      public static final boolean isOneBoardShip(int xCoord, int yCoord, MapSettings map)
      {
         boolean isOneBoardShip = false;

         if (xCoord - 1 >= 0)
         {
            isOneBoardShip = map.Map[xCoord - 1][yCoord] == ECellStatus.LocatedNearShip || map.Map[xCoord -
1][yCoord] == ECellStatus.Empty;
         } else if (xCoord - 1 < 0)
            isOneBoardShip = true;

         if (isOneBoardShip)
            if (xCoord + 1 <= 9)
            {
               isOneBoardShip = map.Map[xCoord + 1][yCoord] == ECellStatus.LocatedNearShip || map.Map[xCoord +
1][yCoord] == ECellStatus.Empty;
            } else if (xCoord + 1 > 9)
               isOneBoardShip = true;

         if (isOneBoardShip)
            if (yCoord - 1 >= 0)
            {
               isOneBoardShip = map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip ||
map.Map[xCoord][yCoord - 1] == ECellStatus.Empty;
            } else if (yCoord - 1 < 0)
               isOneBoardShip = true;
```

```
        if (isOneBoardShip)
            if (yCoord + 1 <= 9)
            {
                isOneBoardShip = map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip ||
map.Map[xCoord][yCoord + 1] == ECellStatus.Empty;
            } else if (yCoord + 1 > 9)
                isOneBoardShip = true;

        return isOneBoardShip;
    }

    public static final char getShipDirection(int xCoord, int yCoord, MapSettings map)
    {
        boolean isHorizontalShip = false;

        if ((xCoord < 9) && (xCoord > 0) && (yCoord < 9) && (yCoord > 0))
        {
            isHorizontalShip = (map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip ||
map.Map[xCoord][yCoord + 1] == ECellStatus.Empty)
                    && (map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord - 1] ==
ECellStatus.Empty);
        } else if (xCoord == 0)
        {
            if (yCoord > 0 && yCoord < 9)
            {
                isHorizontalShip = (map.Map[xCoord + 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord +
1][yCoord] == ECellStatus.Hited)
                        && ((map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord - 1]
== ECellStatus.Empty)
                        && (map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord + 1]
== ECellStatus.Empty));
            } else if (yCoord == 0)
            {
                isHorizontalShip = (map.Map[xCoord + 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord +
1][yCoord] == ECellStatus.Hited)
                        && (map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord + 1]
== ECellStatus.Empty);
            } else if (yCoord == 9)
            {
                isHorizontalShip = (map.Map[xCoord + 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord +
1][yCoord] == ECellStatus.Hited)
                        && (map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord - 1]
== ECellStatus.Empty);
            }
        } else if (xCoord == 9)
        {
            if (yCoord > 0 && yCoord < 9)
            {
                isHorizontalShip = (map.Map[xCoord - 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord -
1][yCoord] == ECellStatus.Hited)
                        && ((map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord - 1]
== ECellStatus.Empty)
                        && (map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord + 1]
== ECellStatus.Empty));
            } else if (yCoord == 0)
            {
                isHorizontalShip = (map.Map[xCoord - 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord -
1][yCoord] == ECellStatus.Hited)
                        && (map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord + 1]
== ECellStatus.Empty);
            } else if (yCoord == 9)
            {
```

```
            isHorizontalShip = (map.Map[xCoord - 1][yCoord] == ECellStatus.ContainsShip || map.Map[xCoord -
1][yCoord] == ECellStatus.Hited)
                    && (map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip || map.Map[xCoord][yCoord - 1]
== ECellStatus.Empty);
            }
        } else if (yCoord == 0)
        {
            isHorizontalShip = map.Map[xCoord][yCoord + 1] == ECellStatus.LocatedNearShip
                || map.Map[xCoord][yCoord + 1] == ECellStatus.Empty;
        } else if (yCoord == 9)
        {
            isHorizontalShip = map.Map[xCoord][yCoord - 1] == ECellStatus.LocatedNearShip
                || map.Map[xCoord][yCoord - 1] == ECellStatus.Empty;
        }

        return (isHorizontalShip == true ? 'h' : 'v');
    }

    public static final boolean isShipDestroyed(int xCoord, int yCoord, MapSettings map, char shipDirection)
    {
        boolean isDestroyed = true;

        if (shipDirection == 'h')
        {
            int offset = 0;

            while ((xCoord + offset <= 9) && map.Map[xCoord + offset][yCoord] != ECellStatus.LocatedNearShip)
            {
                if (map.Map[xCoord + offset][yCoord] == ECellStatus.ContainsShip)
                {
                    isDestroyed = false;
                    break;
                } else
                    offset++;
            }

            offset = 0;

            if (isDestroyed)
            {
                while ((xCoord - offset >= 0) && map.Map[xCoord - offset][yCoord] != ECellStatus.LocatedNearShip)
                {
                    if (map.Map[xCoord - offset][yCoord] == ECellStatus.ContainsShip)
                    {
                        isDestroyed = false;
                        break;
                    } else
                        offset++;
                }
            }

        } else if (shipDirection == 'v') // include empty in alg.
        {
            int offset = 0;

            while ((yCoord + offset <= 9) && map.Map[xCoord][yCoord + offset] != ECellStatus.LocatedNearShip)
            {
                if (map.Map[xCoord][yCoord + offset] == ECellStatus.ContainsShip)
                {
                    isDestroyed = false;
                    break;
                } else
```

```
                offset++;
            }

            offset = 0;

            if (isDestroyed)
            {
                while ((yCoord - offset >= 0) && map.Map[xCoord][yCoord - offset] != ECellStatus.LocatedNearShip)
                {
                    if (map.Map[xCoord][yCoord - offset] == ECellStatus.ContainsShip)
                    {
                        isDestroyed = false;
                        break;
                    } else
                        offset++;
                }
            }
        }

        return isDestroyed;
    }
}
```

Вихідний текст класу MapSettings:

```
package battleship;

public class MapSettings
{
    public ECellStatus[][] Map;

    public MapSettings ()
    {
        this.Map = new ECellStatus[10][10]; // first - x, second - y. You should to use [x][y] when you
                                                                                    // want to request or set data.
    }
}
```

Вихідний текст класу Map:

```
package battleship;

import java.util.Random;

public class Map implements IMap //This is Model in MVP pattern
{

    private MapSettings _userMap;
    private MapSettings _enemyMap;

    @Override
    public String GetMaps(boolean showEnemyShips)
    {
        String res = "";

        for (int i = 0; i < 10; i++)
        {
            for (int j = 0; j < 10; j++)
            {
                if (_userMap.Map[j][i] == ECellStatus.ContainsShip)
                    res += "[S]";
                else if (_userMap.Map[j][i] == ECellStatus.Hited)
                    res += "[X]";
```

```
            else if (_userMap.Map[j][i] == ECellStatus.Empty)
                res += "[*]";
            else
                res += "[ ]";
        }

        res += "   |   ";
        for (int j = 0; j < 10; j++)
        {
            if (_enemyMap.Map[j][i] == ECellStatus.Empty)
                res += "[*]";
            else if (_enemyMap.Map[j][i] == ECellStatus.Hited)
                res += "[X]";
            else if (showEnemyShips && _enemyMap.Map[j][i] == ECellStatus.ContainsShip)
                res += "[S]";
            else
                res += "[ ]";
        }

        res += "\n";
    }

    return res;
}

@Override
public String GetSpecifiedMap(MapSettings ms)
{
    String res = "";

    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            if (ms.Map[j][i] == ECellStatus.ContainsShip)
                res += "[S]";
            else if (ms.Map[j][i] == ECellStatus.LocatedNearShip)
                res += "[-]";
            else
                res += "[ ]";
        }
        res += "\n";
    }

    return res;
}

@Override
public MapSettings GetUserMap()
{
    return this._userMap;
}

@Override
public boolean HitEnemyCell(int x, int y) // returns true if we hit enemy ship
{
    if (x <= 10 && y <= 10)
    {
        x--;
        y--;

        switch (this._enemyMap.Map[x][y])
```

```
        {
            case ContainsShip:
            {
                this._enemyMap.Map[x][y] = ECellStatus.Hited;

                if (MapEngine.isOneBoardShip(x, y, this._enemyMap))
                    MapEngine.SurroundShipWithEmptyCell(x, y, this._enemyMap, 'h');
                else
                    CheckShipOnDistruction(x, y, this._enemyMap);

                return true;
            }
            case ClosedEmpty:
            case LocatedNearShip:
            {
                this._enemyMap.Map[x][y] = ECellStatus.Empty;
                return false;
            }
            case Empty:
                return false;
            case Hited:
                return false;
            default:
                return false;
        }
    } else
        return false;
}

// Total 20 cells supposed to be hited for win
@Override
public boolean CheckVictory(boolean checkPlayerMap)
{
    byte cellsHited = 0;

    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            if (checkPlayerMap)
            {
                if (this._userMap.Map[j][i] == ECellStatus.Hited)
                    cellsHited++;
            } else if (this._enemyMap.Map[j][i] == ECellStatus.Hited)
                cellsHited++;
        }
    }

    return (cellsHited == 20);
}

/**
 * Sets user map for use it in future.
 *
 * @param userMap Defines user map.
 * @return true if seted-up successfuly, or false if not.
 */
@Override
public boolean SetUserMap(MapSettings userMap)
{
    try
    {
```

```
            this._userMap = userMap;
            return true;
        } catch (Exception e)
        {
            return false;
        }
    }

    /**
     * Sets enemy map for use it in future.
     *
     * @param enemyMap Defines enemy map.
     * @return true if seted-up successfuly, or false if not.
     */
    @Override
    public boolean SetEnemyMap(MapSettings enemyMap)
    {
        try
        {
            this._enemyMap = enemyMap;
            return true;
        } catch (Exception e)
        {
            return false;
        }
    }

    private void CheckShipOnDistruction(int x, int y, MapSettings distMap)
    {
        if (MapEngine.getShipDirection(x, y, distMap) == 'h')
        {
            if (MapEngine.isShipDestroyed(x, y, distMap, 'h'))
                MapEngine.SurroundShipWithEmptyCell(x, y, distMap, 'h');
        }
        else if (MapEngine.getShipDirection(x, y, distMap) == 'v')
        {
            if (MapEngine.isShipDestroyed(x, y, distMap, 'v'))
                MapEngine.SurroundShipWithEmptyCell(x, y, distMap, 'v');
        }
    }
}
```

Вихідний текст класу ControlLogic:

```
package battleship;

public class ControlLogic implements IControlLogic
{

    private final IUser       _view;
    private final IMap        _map;
    private final IEnemyLogic _enemy;

    public ControlLogic(IUser _view, IMap _map, IEnemyLogic _enemy)
    {
        this._view  = _view;
        this._map   = _map;
        this._enemy = _enemy;
    }

    @Override
    public void StartTheGame()
    {
```

```
    MapSettings userMap = new MapSettings();
    MapSettings enemyMap;
    boolean restoreGame;

    do
    {
        for (int i = 0; i < 10; i++)
            for (int j = 0; j < 10; j++)
                userMap.Map[i][j] = ECellStatus.ClosedEmpty;
        restoreGame = true;
        userMap     = AskForUserMap();

        if (_map.SetUserMap(userMap))
            _view.DisplayMap(_map.GetSpecifiedMap(userMap));
        else
        {
            _view.ShowMessage("\nЩось пішло не так... Налаштуйте мапу знову.");
            continue;
        }


        enemyMap = _enemy.GenerateEnemyMap();
        if (_map.SetEnemyMap(enemyMap))
        {
            _view.ShowMessage("Противник готовий до бою! Нехай почнеться гра!!!");
            _view.DisplayMap(_map.GetMaps(false));
        }
        else
        {
            _view.ShowMessage("Щось пішло не так... Налаштуйте мапу знову.");
            continue;
        }

        restoreGame = PlayGame();

    } while (restoreGame);
}

private MapSettings AskForUserMap()
{
    MapSettings ms       = new MapSettings();
    byte oneBoardShips   = 0; // Total count 4
    byte twoBoardShips   = 0; // Total count 3
    byte threeBoardShips = 0; // Total count 2
    byte fourBoardShips  = 0; // Total count 1
    String[] userCommand;

    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 10; j++)
            ms.Map[i][j] = ECellStatus.ClosedEmpty;

    while ((oneBoardShips < 4) || (twoBoardShips < 3) || (threeBoardShips < 2)
            || (fourBoardShips < 1))
    {
        _view.ShowMessage("\n====| Встановіть свої кораблі |====");
        _view.DisplayMap(_map.GetSpecifiedMap(ms));
        _view.ShowMessage("Однопалубних кораблів залишилось: " + (4 - oneBoardShips)
                + "\nДвопалубних кораблів залишилось: " + (3 - twoBoardShips)
                + "\nТрипалубних кораблів залишилось: " + (2 - threeBoardShips)
                + "\nЧотирипалубних кораблів залишилось: " + (1 - fourBoardShips));

        userCommand = _view.AskCommand();
```

```
        if (userCommand[0].equals("set"))
        {
           switch (userCommand[3])
           {
              case "1":
              {
                 if (oneBoardShips < 4)
                 {
                    if (MapEngine.SetShipByCoord(ms, Integer.parseInt(userCommand[1]),
Integer.parseInt(userCommand[2]), Integer.parseInt(userCommand[3]), userCommand[4]))
                       oneBoardShips++;
                    else
                       _view.ShowMessage("Хибні параметри команди set. Можливо, ви намагаєтесь встановити
корабель за границями поля"
                             + " або занадто близько до іншого корабля.");
                 } else
                    _view.ShowMessage("Усі доступні кораблі цього типу вже встановлені.");
                 break;
              }
              case "2":
              {
                 if (twoBoardShips < 3)
                 {
                    if (MapEngine.SetShipByCoord(ms, Integer.parseInt(userCommand[1]),
Integer.parseInt(userCommand[2]), Integer.parseInt(userCommand[3]), userCommand[4]))
                       twoBoardShips++;
                    else
                       _view.ShowMessage("Хибні параметри команди set. Можливо, ви намагаєтесь встановити
корабель за границями поля"
                             + " або занадто близько до іншого корабля.");
                 } else
                    _view.ShowMessage("Усі доступні кораблі цього типу вже встановлені.");
                 break;
              }
              case "3":
              {
                 if (threeBoardShips < 2)
                 {
                    if (MapEngine.SetShipByCoord(ms, Integer.parseInt(userCommand[1]),
Integer.parseInt(userCommand[2]), Integer.parseInt(userCommand[3]), userCommand[4]))
                       threeBoardShips++;
                    else
                       _view.ShowMessage("Хибні параметри команди set. Можливо, ви намагаєтесь встановити
корабель за границями поля"
                             + " або занадто близько до іншого корабля.");
                 } else
                    _view.ShowMessage("Усі доступні кораблі цього типу вже встановлені.");
                 break;
              }
              case "4":
              {
                 if (fourBoardShips < 1)
                 {
                    if (MapEngine.SetShipByCoord(ms, Integer.parseInt(userCommand[1]),
Integer.parseInt(userCommand[2]), Integer.parseInt(userCommand[3]), userCommand[4]))
                       fourBoardShips++;
                    else
                       _view.ShowMessage("Хибні параметри команди set. Можливо, ви намагаєтесь встановити
корабель за границями поля"
                             + " або занадто близько до іншого корабля.");
                 } else
                    _view.ShowMessage("Усі доступні кораблі цього типу вже встановлені.");
```

```
                    break;
                }
            }
        } else if (userCommand[0].equals("reset"))
        {
            oneBoardShips   = 0;
            twoBoardShips   = 0;
            threeBoardShips = 0;
            fourBoardShips  = 0;

            for (int i = 0; i < 10; i++)
                for (int j = 0; j < 10; j++)
                    ms.Map[i][j] = ECellStatus.ClosedEmpty;

            _view.ShowMessage("Налаштування мапи та прогрес були скинуті.");
        } else
            _view.ShowMessage("Хибна команда. Вам необхідно встановити поле перед початком гри.");
    }

    return ms;
  }

  private boolean PlayGame() // Returns "true" if game was restored
  {
    boolean isVictory     = false; // true if User have won
    boolean isRestored    = false;
    boolean isStillInGame = true; // false if PC have won
    boolean isUserTurn    = true;
    String[] userCommand;

    do
    {
      do
      {
        userCommand = _view.AskCommand();

        switch (userCommand[0])
        {
          case "hit":
          {
            isUserTurn = _map.HitEnemyCell(Integer.parseInt(userCommand[1]),
Integer.parseInt(userCommand[2]));
            isVictory  = _map.CheckVictory(false);

            _view.DisplayMap(_map.GetMaps(false));

            break;
          }
          case "reset":
          {
            isUserTurn = false;
            isRestored = true;

            break;
          }
          default:
            _view.ShowMessage("Мапу вже налаштовано, ви не можете встановити корабель. Щоб почати
спочатку, введіть \"reset\"");
            break;
        }

      } while (isUserTurn && !isVictory);
```

```
            if (!isRestored && !isVictory)
            {
                _view.ShowMessage("Зачекайте, ходить комп'ютер...");
                _enemy.Fire(_map.GetUserMap());
                _view.DisplayMap(_map.GetMaps(false));
                isStillInGame = !_map.CheckVictory(true);
            }

        } while (!isVictory && !isRestored && isStillInGame);

        if (!isStillInGame)
        {
            _view.ShowMessage("Комп'ютер виграв!!! \nОсь, де були кораблі супротивника:");
            _view.DisplayMap(_map.GetMaps(true));
        }
        if (isVictory)
            _view.ShowMessage("Ви виграли!!!");
        if (isRestored)
            _view.ShowMessage("Налаштування мапи та прогрес були скинуті.\n");

        return isRestored;
    }

}
```

Вихідний текст класу EnemyLogic:

```
package battleship;

import java.util.Random;

public class EnemyLogic implements IEnemyLogic
{
    class EnemyFireHistory
    {
        private int _LastHitedX = -1;
        private int _LastHitedY = -1;
        private boolean _findVertical     = true;
        private boolean _isLeftCellChecked  = false;
        private boolean _isRightCellChecked = false;
        private boolean _isUpCellChecked    = false;
        private boolean _isDownCellChecked  = false;
    }

    @Override
    // MapSettings (userMap) ALWAYS WILL BE PASSED BY REFERENCE!!!!!
    public void Fire(MapSettings userMap)
    {
        Random rand = new Random();
        EnemyFireHistory history = new EnemyFireHistory();
        boolean success = false;
        boolean moveDone = false;
        byte xCoord;
        byte yCoord;


        if (history._LastHitedX == -1 && history._LastHitedY == -1)
        {
            while (!success)
            {
                xCoord = (byte) (rand.nextInt(10));
                yCoord = (byte) (rand.nextInt(10));
```

```
            if (userMap.Map[xCoord][yCoord] == ECellStatus.Hited || userMap.Map[xCoord][yCoord] ==
ECellStatus.Empty)
                success = false;
            else
            {
                if (userMap.Map[xCoord][yCoord] == ECellStatus.ContainsShip)
                {
                    userMap.Map[xCoord][yCoord] = ECellStatus.Hited;
                    success = true;

                    if (!MapEngine.isOneBoardShip(xCoord, yCoord, userMap))
                    {
                        history._LastHitedX = xCoord;
                        history._LastHitedY = yCoord;
                    }
                    else
                    {
                        MapEngine.SurroundShipWithEmptyCell(xCoord, yCoord, userMap, 'h');
                        success = false;
                    }
                }
                else if (userMap.Map[xCoord][yCoord] == ECellStatus.ClosedEmpty || userMap.Map[xCoord][yCoord] ==
ECellStatus.LocatedNearShip)
                {
                    userMap.Map[xCoord][yCoord] = ECellStatus.Empty;
                    success = true;
                }
            }
        }
    }



    if (history._LastHitedX != -1 && history._LastHitedY != -1)
    {
        if (!history._isLeftCellChecked && (history._LastHitedX - 1 >= 0))
        {
            for (int i = history._LastHitedX - 1; i >= history._LastHitedX - 4; i--)
            {
                if (i >= 0 && (userMap.Map[i][history._LastHitedY] == ECellStatus.ContainsShip))
                {
                    userMap.Map[i][history._LastHitedY] = ECellStatus.Hited;
                    history._findVertical = false;
                }
                else
                {
                    if (i >= 0 && userMap.Map[i][history._LastHitedY] == ECellStatus.LocatedNearShip &&
!MapEngine.isShipDestroyed(i, history._LastHitedY, userMap, 'h'))
                    {
                        userMap.Map[i][history._LastHitedY] = ECellStatus.Empty;
                        moveDone = true;
                    }
                    else if (i >= 0 && MapEngine.isShipDestroyed(i, history._LastHitedY, userMap, 'h'))
                    {
                        history._isRightCellChecked = true;
                        moveDone = false;
                    }

                    history._isLeftCellChecked = true;
                    break;
                }
```

```
            }

            if (history._LastHitedX + 1 <= 9)
            {
               if (userMap.Map[history._LastHitedX + 1][history._LastHitedY] == ECellStatus.Empty)
                  history._isRightCellChecked = true;
               else if (userMap.Map[history._LastHitedX + 1][history._LastHitedY] == ECellStatus.LocatedNearShip)
                  history._isRightCellChecked = true;
            }
         }
         else if (history._LastHitedX - 1 < 0)
            history._isLeftCellChecked = true;



         if (!history._isRightCellChecked && (history._LastHitedX + 1 <= 9) && !moveDone)
         {
            for (int i = history._LastHitedX + 1; i <= history._LastHitedX + 4; i++)
            {
               if (i <= 9 && (userMap.Map[i][history._LastHitedY] == ECellStatus.ContainsShip))
               {
                  userMap.Map[i][history._LastHitedY] = ECellStatus.Hited;
                  history._findVertical = false;
               }
               else
               {
                  if (i <= 9 && userMap.Map[i][history._LastHitedY] == ECellStatus.LocatedNearShip &&
!MapEngine.isShipDestroyed(i, history._LastHitedY, userMap, 'h'))
                  {
                     userMap.Map[i][history._LastHitedY] = ECellStatus.Empty;
                     moveDone = true;
                  }
                  else if (i <= 9 && MapEngine.isShipDestroyed(i, history._LastHitedY, userMap, 'h'))
                     moveDone = false;


                  history._isRightCellChecked = true;
                  break;
               }
            }
         }
         else if (history._LastHitedX + 1 > 9)
            history._isRightCellChecked = true;



         if (history._findVertical == true) // fix down direction
         {
            if (!history._isUpCellChecked && (history._LastHitedY - 1 >= 0) && !moveDone)
            {
               for (int i = history._LastHitedY - 1; i >= history._LastHitedY - 4; i--)
               {
                  if (i >= 0 && (userMap.Map[history._LastHitedX][i] == ECellStatus.ContainsShip))
                  {
                     userMap.Map[history._LastHitedX][i] = ECellStatus.Hited;
                  }
                  else
                  {
                     if (i >= 0 && userMap.Map[history._LastHitedX][i] == ECellStatus.LocatedNearShip &&
!MapEngine.isShipDestroyed(history._LastHitedX, i, userMap, 'v'))
                     {
                        userMap.Map[history._LastHitedX][i] = ECellStatus.Empty;
```

```
                    moveDone = true;
                }
                else if (i >= 0 && MapEngine.isShipDestroyed(history._LastHitedX, i, userMap, 'v'))
                {
                    history._isDownCellChecked = true;
                    moveDone = false;
                }

                history._isUpCellChecked = true;
                break;
            }
        }

        if (history._LastHitedY + 1 <= 9)
        {
            if (userMap.Map[history._LastHitedX][history._LastHitedY + 1] == ECellStatus.Empty
                || userMap.Map[history._LastHitedX][history._LastHitedY + 1] == ECellStatus.LocatedNearShip)
                history._isDownCellChecked = true;
        }

    }
    else if (history._LastHitedY - 1 < 0)
        history._isUpCellChecked = true;

    if (!history._isDownCellChecked && (history._LastHitedY + 1 <= 9) && !moveDone)
    {
        for (int i = history._LastHitedY + 1; i <= history._LastHitedY + 4; i++)
        {
            if (i <= 9 && (userMap.Map[history._LastHitedX][i] == ECellStatus.ContainsShip))
            {
                userMap.Map[history._LastHitedX][i] = ECellStatus.Hited;
            }
            else
            {
                if (i <= 9 && userMap.Map[history._LastHitedX][i] == ECellStatus.LocatedNearShip &&
!MapEngine.isShipDestroyed(history._LastHitedX, i, userMap, 'v'))
                {
                    userMap.Map[history._LastHitedX][i] = ECellStatus.Empty;
                    moveDone = true;
                }
                else if (i <= 9 && MapEngine.isShipDestroyed(history._LastHitedX, i, userMap, 'v'))
                    moveDone = false;

                history._isDownCellChecked = true;
                break;
            }
        }
    }
    else if (history._LastHitedY + 1 > 9)
        history._isDownCellChecked = true;
}

if (history._isLeftCellChecked && history._isRightCellChecked && !history._findVertical)
{
    MapEngine.SurroundShipWithEmptyCell(history._LastHitedX, history._LastHitedY, userMap, 'h');

    history._LastHitedX = -1;
    history._LastHitedY = -1;

    history._isLeftCellChecked  = false;
    history._isRightCellChecked = false;
    history._isUpCellChecked    = false;
```

```
            history._isDownCellChecked  = false;
            history._findVertical       = true;

            if (!moveDone)
               this.Fire(userMap);
         }
         else if (history._isDownCellChecked && history._isUpCellChecked)
         {
            MapEngine.SurroundShipWithEmptyCell(history._LastHitedX, history._LastHitedY, userMap, 'v');

            history._LastHitedX = -1;
            history._LastHitedY = -1;

            history._isLeftCellChecked  = false;
            history._isRightCellChecked = false;
            history._isUpCellChecked    = false;
            history._isDownCellChecked  = false;
            history._findVertical       = true;

            if (!moveDone)
               this.Fire(userMap);
         }
      }
   }

   @Override
   public MapSettings GenerateEnemyMap()
   {
      MapSettings ms = new MapSettings();
      Random rand    = new Random();
      byte oneBoardShips   = 0; // Total count 4.
      byte twoBoardShips   = 0; // Total count 3.
      byte threeBoardShips = 0; // Total count 2.
      byte fourBoardShips  = 0; // Total count 1.
      byte xCoord; // For ship set-up. X coordinate of ship [1..10].
      byte yCoord; // For ship set-up. Y coordinate of ship [1..10].
      byte boardNumber; // For ship set-up. Boards number [1..4].
      String direction; // For ship set-up. Ship direction l, r, u or d.

      for (int i = 0; i<10; i++)
         for (int j = 0; j<10; j++)
            ms.Map[i][j] = ECellStatus.ClosedEmpty;

      while ((oneBoardShips < 4) || (twoBoardShips < 3) || (threeBoardShips < 2)
            || (fourBoardShips < 1))
      {
         boardNumber = (byte) (rand.nextInt(4) + 1);
         xCoord      = (byte) (rand.nextInt(10) + 1);
         yCoord      = (byte) (rand.nextInt(10) + 1);

         switch (rand.nextInt(4))
         {
            case 0:
               direction = "l";
               break;
            case 1:
               direction = "r";
               break;
            case 2:
               direction = "u";
               break;
            case 3:
```

```
                direction = "d";
                break;
            default:
                direction = "l";
                break;
        }

        switch (boardNumber)
        {
            case 1:
                if (oneBoardShips < 4)
                    if (MapEngine.SetShipByCoord(ms, xCoord, yCoord, boardNumber, direction))
                        oneBoardShips++;
                break;
            case 2:
                if (twoBoardShips < 3)
                    if (MapEngine.SetShipByCoord(ms, xCoord, yCoord, boardNumber, direction))
                        twoBoardShips++;
                break;
            case 3:
                if (threeBoardShips < 2)
                    if (MapEngine.SetShipByCoord(ms, xCoord, yCoord, boardNumber, direction))
                        threeBoardShips++;
                break;
            case 4:
                if (fourBoardShips < 1)
                    if (MapEngine.SetShipByCoord(ms, xCoord, yCoord, boardNumber, direction))
                        fourBoardShips++;
                break;
        }
    }

    return ms;
  }
}
```

Вихідний текст класу Battleship:

```
package battleship;

public class Battleship
{
  public static void main(String[] args)
  {
    IMap map            = new Map();
    IUser user          = new UserInterface();
    IEnemyLogic enemy   = new EnemyLogic();
    IControlLogic logic = new ControlLogic(user, map, enemy);

    logic.StartTheGame();
  }
}
```

ВИСНОВОК

На основі ескізного проекту було створено гру «Морський бій», котра повністю відповідає необхідним вимогам. Програма легка для використання, а алгоритм гри супротивника досить непагано імітує гру людини. Таким чином, можна сказати, що програма виконує поставлені цілі. Отже, роботу виконано успішно.