Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность 09.02.07 Информационные системы и программирование

Квалификация: Программист

Практическая работа №4

По дисциплине «МДК 04.01 Внедрение и поддержка компьютерных систем»

Листов:

|  |  |
| --- | --- |
| Выполнила:  Петриашвили М.З.  Студент группы: П50-6-22 | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_А.А. Никонова  «\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 года |

Москва 2025

Цель работы: разработать консольную игру «Морской бой» на языке программирования Dart в среде Visual Studio Code с реализацией случайной расстановки кораблей, отображением игрового поля, пошаговым процессом игры, проверкой корректности ходов и определением победителя.

Ход работы:

В данной практической работе были реализованы следующие методы, выполняющие свои определенные функции:

### Класс **Cell**

* Описывает одну клетку поля.
* ship — есть ли в клетке корабль.
* hit — была ли клетка поражена (попадание).
* miss — промах.

### Класс **Board** (Игровое поле)

* Хранит матрицу клеток 10×10.

Методы:

* display({bool hideShips = false})  
  Отображает поле.
  + X — попадание
  + • — промах
  + O — корабль (если hideShips = false)
  + ~ — пустая клетка
* placeShip(int x, int y, int length, bool horizontal)  
  Размещает корабль длиной length на поле.
  + Проверяет, чтобы корабль не выходил за границы и не пересекался с другими.
  + Возвращает true, если удалось разместить.
* shoot(int x, int y)  
  Выполняет выстрел по координатам.
  + Если попали в корабль → отмечает hit.
  + Если мимо → отмечает miss.
  + Возвращает true при попадании.
* allShipsDestroyed()  
  Проверяет, остались ли у игрока целые корабли.
  + Если все уничтожены → возвращает true.

### Класс **Game**

* Хранит два игровых поля (игрок и компьютер).
* vsAI — флаг режима против компьютера.
* rand — генератор случайных чисел.

Методы:

* randomPlacement(Board board)  
  Случайным образом расставляет корабли на поле.
  + Список кораблей: [4, 3, 3, 2, 2, 2, 1, 1, 1, 1].
  + Корабли могут быть горизонтальными и вертикальными.
* play()  
  Основной игровой цикл.
  + Расставляет корабли игрока и компьютера.
  + Отображает поля.
  + Реализует пошаговый процесс игры:
    - Игрок вводит координаты (например, A5).
    - Компьютер стреляет случайным образом.
  + Проверяет победу после каждого хода.
  + В конце предлагает сыграть заново.

### Функция **main()**

* Точка входа в программу.
* Отображает меню:
  + **1** — игра против компьютера
  + **2** — игра против другого игрока (пока используется ИИ)
* Запускает игровой процесс.

Код работы:

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность 09.02.07 Информационные системы и программирование

Квалификация: Программист

Практическая работа №2

По дисциплине «МДК 04.01 Внедрение и поддержка компьютерных систем»

Листов:

|  |  |
| --- | --- |
| Выполнила:  Петриашвили М.З.  Студент группы: П50-6-22 | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_А.А. Никонова  «\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 года |

Москва 2025

Цель работы: разработать консольную игру «Морской бой» на языке программирования Dart в среде Visual Studio Code с реализацией случайной расстановки кораблей, отображением игрового поля, пошаговым процессом игры, проверкой корректности ходов и определением победителя.

Ход работы:

В данной практической работе были реализованы следующие методы, выполняющие свои определенные функции:

### Основные классы и их назначение

#### **Класс** Cell

Определяет состояние одной клетки игрового поля.

Содержит три логических поля:

* ship — в клетке находится часть корабля;
* hit — по этой клетке был произведён выстрел и корабль поражён;
* miss — выстрел был сделан, но мимо.

#### **Класс** Board

Представляет игровое поле размером 10×10 клеток.

Основные методы:

* display({bool hideShips = false}) — выводит поле на экран.
  + ~ — неизвестная клетка;
  + O — корабль (если не скрыт);
  + X — попадание;
  + • — промах.
* placeShip(x, y, length, horizontal) — пытается разместить корабль указанной длины на поле (горизонтально или вертикально). Возвращает true, если размещение успешно.
* shoot(x, y) — выполняет выстрел по клетке. Возвращает true, если попадание.
* allShipsDestroyed() — проверяет, уничтожены ли все корабли.
* countRemainingShips() и countDestroyedShips() — подсчитывают количество целых и поражённых палуб соответственно.

#### **Класс** PlayerStats

Собирает статистику игрока:

* hits — количество попаданий;
* misses — количество промахов;
* turns — количество ходов;
* totalShots — общее число выстрелов (геттер).

#### **Класс** Game

Основной управляющий класс, реализующий игровую логику.  
Содержит два поля:

* player1 — доска первого игрока;
* player2 — доска второго игрока или компьютера.

Дополнительные элементы:

* randomPlacement(board) — случайно размещает корабли.
* manualPlacement(board) — ручное размещение с вводом координат.
* parseInput(input) — преобразует ввод пользователя (например, A5) в координаты [x, y].
* saveStats(winner) — сохраняет результаты игры в текстовый файл stats/game\_stats.txt.
* play() — основной цикл игры.

### 3. Игровой процесс

1. После запуска отображается главное меню:
   * выбрать режим:  
     1—против компьютера;  
     2 — против игрока.
2. Далее выбирается способ расстановки кораблей:
   * 1 — случайно;
   * 2 — вручную.
3. Игроки делают выстрелы, вводя координаты (например, A5 → строка A, столбец 5).
4. После каждого хода отображаются оба поля:
   * поле игрока с кораблями;
   * поле противника (корабли скрыты).
5. Игра продолжается до полного уничтожения всех кораблей одного игрока.

### 4. Формат сохранения статистики

После окончания игры создается папка stats и файл game\_stats.txt, в котором фиксируются:

Результат игры:

Победитель: Игрок 1

---

Игрок 1:

Попаданий: 15

Промахов: 22

Всего выстрелов: 37

Уничтожено палуб противника: 20

Осталось своих палуб: 5/20

Количество ходов: 37

Компьютер:

Попаданий: 20

Промахов: 18

Всего выстрелов: 38

Уничтожено палуб противника: 15

Осталось своих палуб: 0/20

Количество ходов: 38

Код работы:

import 'dart:io';

import 'dart:math';

const int size = 10;

class Cell {

  bool ship = false;

  bool hit = false;

  bool miss = false;

}

class Board {

  List<List<Cell>> grid =

      List.generate(size, (\_) => List.generate(size, (\_) => Cell()));

  void display({bool hideShips = false}) {

    stdout.write("   ");

    for (int i = 0; i < size; i++) stdout.write("$i ");

    print("");

    for (int y = 0; y < size; y++) {

      stdout.write("${String.fromCharCode(65 + y)}  ");

      for (int x = 0; x < size; x++) {

        Cell c = grid[y][x];

        if (c.hit) {

          stdout.write("X ");

        } else if (c.miss) {

          stdout.write("• ");

        } else if (c.ship && !hideShips) {

          stdout.write("O ");

        } else {

          stdout.write("~ ");

        }

      }

      print("");

    }

  }

  bool placeShip(int x, int y, int length, bool horizontal) {

    if (horizontal) {

      if (x + length > size) return false;

      for (int i = 0; i < length; i++) {

        if (grid[y][x + i].ship) return false;

      }

      for (int i = 0; i < length; i++) {

        grid[y][x + i].ship = true;

      }

    } else {

      if (y + length > size) return false;

      for (int i = 0; i < length; i++) {

        if (grid[y + i][x].ship) return false;

      }

      for (int i = 0; i < length; i++) {

        grid[y + i][x].ship = true;

      }

    }

    return true;

  }

  bool shoot(int x, int y) {

    if (grid[y][x].hit || grid[y][x].miss) {

      return false;

    }

    if (grid[y][x].ship) {

      grid[y][x].hit = true;

      return true;

    } else {

      grid[y][x].miss = true;

      return false;

    }

  }

  bool allShipsDestroyed() {

    for (var row in grid) {

      for (var cell in row) {

        if (cell.ship && !cell.hit) return false;

      }

    }

    return true;

  }

}

class Game {

  Board player1 = Board();

  Board player2 = Board();

  bool vsAI;

  Random rand = Random();

  static int playerWins = 0;

  static int aiWins = 0;

  Game(this.vsAI);

  void randomPlacement(Board board) {

    List<int> ships = [4, 3, 3, 2, 2, 2, 1, 1, 1, 1];

    for (var ship in ships) {

      bool placed = false;

      while (!placed) {

        int x = rand.nextInt(size);

        int y = rand.nextInt(size);

        bool horizontal = rand.nextBool();

        placed = board.placeShip(x, y, ship, horizontal);

      }

    }

  }

  void manualPlacement(Board board) {

    List<int> ships = [4, 3, 3, 2, 2, 2, 1, 1, 1, 1];

    for (var ship in ships) {

      bool placed = false;

      while (!placed) {

        print("Введите координаты для корабля длиной $ship (например, A5):");

        String? input = stdin.readLineSync();

        print("Введите направление (h - горизонтально, v - вертикально):");

        String? dir = stdin.readLineSync();

        if (input == null || input.length < 2) {

          print("Ошибка: неправильный ввод!");

          continue;

        }

        int y = input.codeUnitAt(0) - 65;

        int? x = int.tryParse(input.substring(1));

        bool horizontal = (dir?.toLowerCase() == "h");

        if (x == null || x < 0 || x >= size || y < 0 || y >= size) {

          print("Ошибка: неверные координаты!");

          continue;

        }

        placed = board.placeShip(x, y, ship, horizontal);

        if (!placed) {

          print("Ошибка: нельзя разместить корабль здесь!");

        } else {

          board.display();

        }

      }

    }

  }

  List<int>? parseInput(String? input) {

    if (input == null || input.length < 2) return null;

    int y = input.codeUnitAt(0) - 65;

    int? x = int.tryParse(input.substring(1));

    if (x == null || x < 0 || x >= size || y < 0 || y >= size) return null;

    return [x, y];

  }

  void play({bool manual = false}) {

    if (manual) {

      print("Ручная расстановка кораблей для Игрока 1:");

      manualPlacement(player1);

    } else {

      randomPlacement(player1);

    }

    randomPlacement(player2);

    bool turn = true;

    while (true) {

      print("\nПоле игрока 1:");

      player1.display();

      print("\nПоле соперника:");

      player2.display(hideShips: true);

      if (turn) {

        print("\nИгрок 1, ваш ход (например, A5):");

        String? input = stdin.readLineSync();

        var coords = parseInput(input);

        if (coords == null) {

          print("Ошибка: неверный ввод!");

          continue;

        }

        bool hit = player2.shoot(coords[0], coords[1]);

        print(hit ? "Игрок 1 попал!" : "Игрок 1 промахнулся!");

        if (player2.allShipsDestroyed()) {

          print("Игрок 1 победил!");

          playerWins++;

          break;

        }

      } else {

        if (vsAI) {

          print("\nХод компьютера...");

          int x, y;

          do {

            x = rand.nextInt(size);

            y = rand.nextInt(size);

          } while (player1.grid[y][x].hit || player1.grid[y][x].miss);

          bool hit = player1.shoot(x, y);

          print(

              "Компьютер стреляет в ${String.fromCharCode(65 + y)}$x и ${hit ? "попадает!" : "промахивается!"}");

          if (player1.allShipsDestroyed()) {

            print("Компьютер победил!");

            aiWins++;

            break;

          }

        } else {

          print("\nИгрок 2, ваш ход (например, B7):");

          String? input = stdin.readLineSync();

          var coords = parseInput(input);

          if (coords == null) {

            print("Ошибка: неверный ввод!");

            continue;

          }

          bool hit = player1.shoot(coords[0], coords[1]);

          print(hit ? "Игрок 2 попал!" : "Игрок 2 промахнулся!");

          if (player1.allShipsDestroyed()) {

            print("Игрок 2 победил!");

            aiWins++;

            break;

          }

        }

      }

      turn = !turn;

    }

    print("\nТекущий счёт: Игрок 1 – $playerWins | Соперник – $aiWins");

    print("\nХотите сыграть ещё раз? (y/n)");

    if (stdin.readLineSync()?.toLowerCase() == "y") {

      Game(vsAI).play(manual: manual);

    }

  }

}

void main() {

  print("Морской бой");

  print("1. Играть против компьютера");

  print("2. Играть против другого игрока");

  String? choice = stdin.readLineSync();

  print("Выберите способ расстановки кораблей:");

  print("1. Случайная");

  print("2. Ручная");

  String? setup = stdin.readLineSync();

  Game(choice == "1").play(manual: setup == "2");

}

Вывод: в ходе работы была реализована консольная версия игры «Морской бой» на языке Dart. Программа позволяет играть против компьютера, поддерживает случайную расстановку кораблей, ввод и обработку координат, отображение игрового поля, а также проверку победителя. Таким образом, поставленная цель работы достигнута.

Вывод: в ходе работы была реализована консольная версия игры «Морской бой» на языке Dart. Программа позволяет играть против компьютера, поддерживает случайную расстановку кораблей, ввод и обработку координат, отображение игрового поля, а также проверку победителя. Таким образом, поставленная цель работы достигнута.