Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

По курсу: "Архитектура ЭВМ"

Студент Сукочева Алис
Группа ИУ7-53Б
Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема Создание хранилищ. Создание классов. Таймер.

Студент: _____ Сукочева А.
подпись, дата Фамилия, И.О.

Преподаватель: _____ Попов А. Ю.
подпись, дата Фамилия, И. О.

Москва — 2020 г.

TASK_1. Задание 1

Техническое задание:

— Создать хранилище в оперативной памяти для хранения информации о детях.

— Необходимо хранить информацию о ребенке: фамилия и возраст.

— Необходимо обеспечить уникальность фамилий детей.

Реализовать функции:

— CREATE READ UPDATE DELETE для детей в хранилище

— Получение среднего возраста детей

— Получение информации о самом старшем ребенке

— Получение информации о детях, возраст которых входит в заданный отрезок

— Получение информации о детях, фамилия которых начинается с заданной буквы

— Получение информации о детях, фамилия которых длиннее заданного количества символов

— Получение информации о детях, фамилия которых начинается с гласной буквы

Листинг 1 — Код программы. Задание 1.

```
1  "use strict";
2
3  class Kids {
4      constructor() {
5          this.arr = [];
6      }
7
8      add(surname, age) {
9          // function func(elem) {
10         //  return elem.surname === surname;
11         // }
12         if (!this.arr.find(elem => elem.surname == surname)) {
13             let new_kid = { surname, age }; // { surname: surname, age :
                   age }
14             this.arr.push(new_kid);
15         }
16     }
17
18     log() {
19         for (let i = 0; i < this.arr.length; i++)
20             console.log(this.arr[i].surname, " ", this.arr[i].age);
21         console.log("\n")
```

```
22      }
23
24      read(surname) {
25          return this.arr.find(elem => elem.surname === surname);
26          // for (let i = 0; i < this.arr.length; i++)
27          //  if (this.arr[i].surname === surname)
28          //      return this.arr[i];
29          // return ""
30      }
31
32      update(surname, new_age, new_surname = false) {
33          for (let i = 0; i < this.arr.length; i++) {
34              if (this.arr[i].surname === surname) {
35                  this.arr[i].age = new_age;
36                  if (new_surname && !this.arr.find(elem => elem.surname ===
                        new_surname))
37                      this.arr[i].surname = new_surname;
38                  return;
39              }
40          }
41      }
42
43      delete(surname) {
44          this.arr = this.arr.filter(elem =>
45              elem.surname !== surname);
46      }
47
48      get_avg() {
49          let sum = 0;
50          let len = this.arr.length;
51          for (let i = 0; i < len; i++)
52              sum += this.arr[i].age;
53          return len ? sum / len : 0;
54      }
55
56      get_eldest() {
57          let len = this.arr.length;
58          if (!len)
59              return;
60          let max = this.arr[0];
61          for (let i = 1; i < len; i++) {
62              if (this.arr[i].age > max.age)
63                  max = this.arr[i];
64          }
65          return max;
66      }
67
```

```
    get_age_range(begin, end) {
        return this.arr.filter(elem => elem.age >= begin
            && elem.age <= end);
    }

    get_surname_letter(letter) {
        return this.arr.filter(elem => elem.surname.charAt(0) === letter);
    }

    get_surname_len(len) {
        return this.arr.filter(elem => elem.surname.length > len)
    }

    get_surname_volwels() {
        const VOWELS = ['A', 'E', 'I', 'O', 'U'];
        return this.arr.filter(elem =>
            VOWELS.indexOf(elem.surname.charAt(0).toUpperCase()) != -1)
    }
};
```

Листинг 2 — Код тестов. Задание 1.

```
function test_kids() {
    let child = new Kids();
    child.add("Sukocheva", 2);
    child.add("Sukocheva", 3);
    child.add("Namestnik", 1);
    child.add("Vinogradov", 4);
    child.add("Volkov", 3);
    child.add("Orbitov", 7);


    child.log();

    console.log(child.read("Sukocheva"));
    child.update("Sukocheva", 4)
    console.log(child.read("Sukocheva"));

    child.log();
    child.delete("Vinogradov");
    child.log();

    child.read("Sukocheva")

    console.log(child.get_avg());
    console.log(child.get_eldest())

```

```
26      console.log(child.get_age_range(1, 3))
27      console.log(child.get_surname_letter('S'));
28
29      console.log(child.get_surname_len(6));
30
31      console.log(child.get_surname_volwels());
32  }
```

TASK_1. Задание 2

Техническое задание:

— Создать хранилище в оперативной памяти для хранения информации о студентах.

— Необходимо хранить информацию о студенте: название группы, номер студенческого билета, оценки по программированию.

— Необходимо обеспечить уникальность номеров студенческих билетов.

Реализовать функции:

— CREATE READ UPDATE DELETE для студентов в хранилище

— Получение средней оценки заданного студента

— Получение информации о студентах в заданной группе

— Получение студента, у которого наибольшее количество оценок в заданной группе

— Получение студента, у которого нет оценок

Листинг 3 — Код программы. Задание 1.

```
"use strict";

    class Students {
        constructor() {
            this.arr = [];
        }

        add(id, group, arr_score) {
            if (!this.arr.find(elem => elem.id === id)) {
                let new_student = { group, id, arr_score };
                this.arr.push(new_student);
            }
        }

        log() {
            console.log(this.arr);
        }

        read(id) {
            return this.arr.find(elem => elem.id === id);
        }

        update(id, group, arr_score) {
            let student = this.read(id);
            if (student) {
```

```
            student.group = group;
            student.arr_score = arr_score;
        }
    }

    delete(id) {
        this.arr = this.arr.filter(elem =>
            elem.id !== id);
    }

    get_avg(id) {
        let student = this.read(id);
        if (!student)
            return;

        let len = student.arr_score.length;
        let sum = 0;
        for (let i = 0; i < len; i++)
            sum += student.arr_score[i];

        return len ? sum / len : 0;
    }

    get_info_group(group) {
        return this.arr.filter(elem => elem.group === group);
    }

    get_student_max_count_score(group) {
        let students = this.get_info_group(group);
        if (!students)
            return;
        let max = students[0];
        for (let i = 1; i < students.length; i++) {
            if (students[i].arr_score.length > max.arr_score.length)
                max = students[i];
        }
        return max
    }

    get_student_no_score() {
        return this.arr.filter(elem => elem.arr_score.length === 0);
    }
};
```

Листинг 4 — Код тестов. Задание 2.

```
function test_students() {
```

```
        let students = new Students();

        students.add(123456, "UI7-43b", [5, 5, 5]);
        students.add(123456, "UI7-43b", [5, 5, 5]);

        students.add(123457, "UI7-43b", []);
        students.add(123451, "UI7-43b", [1, 5, 5, 2]);
        students.add(123000, "UI7-44b", [4, 3, 4]);
        students.add(123444, "UI7-45b", [5, 4, 5]);
        students.add(123442, "UI7-45b", [2, 3, 3]);
        students.add(123441, "UI7-42b", [2, 3, 5]);
        students.add(123443, "UI7-41b", [5, 2, 3]);

        students.log();

        console.log(students.read(123456));

        students.update(123441, "UI7-42b", [5, 5, 5]);
        students.log();

        students.delete(123442);
        students.log();

        console.log(students.get_avg(123441));
        console.log(students.get_info_group("UI7-43b"));

        console.log(students.get_student_max_count_score("UI7-43b"))
        console.log(students.get_student_no_score());
    }
```

TASK_1. Задание 3

Техническое задание:

— Создать хранилище в оперативной памяти для хранения точек.

— Неоходимо хранить информацию о точке: имя точки, позиция X и позиция Y.

— Необходимо обеспечить уникальность имен точек.

Реализовать функции:

— CREATE READ UPDATE DELETE для точек в хранилище

— Получение двух точек, между которыми наибольшее расстояние

— Получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную константу

— Получение точек, находящихся выше / ниже / правее / левее заданной оси координат

— Получение точек, входящих внутрь заданной прямоугольной зоны

Листинг 5 — Код программы. Задание 3

```
"use strict";

    class Points {
        constructor() {
            this.arr = [];
        }

        add(name_point, x, y) {
            if (!this.arr.find(elem => elem.name_point === name_point)) {
                let new_point = { name_point, x, y };
                this.arr.push(new_point);
            }
        }

        log() {
            console.log(this.arr);
        }

        read(name_point) {
            return this.arr.find(elem => elem.name_point === name_point);
        }

        update(name_point, x, y) {
            let point = this.read(name_point);
            if (point) {
```

```
26                point.x = x;
27                point.y = y;
28            }
29        }
30
31        delete(name_point) {
32            this.arr = this.arr.filter(elem =>
33                elem.name_point !== name_point);
34        }
35
36        get_distance(p1, p2) {
37            let dx = p1.x - p2.x;
38            let dy = p1.y - p2.y;
39            return Math.sqrt(dx * dx + dy * dy);
40        }
41
42        min_distance() {
43            if (this.arr.length < 2)
44                return;
45
46            let len = this.arr.length;
47            let p1 = this.arr[0];
48            let p2 = this.arr[1];
49            let min_dist = this.get_distance(p1, p2), current_dist;
50            console.log(min_dist);
51
52            for (let i = 0; i < len - 1; i++)
53                for (let j = i + 1; j < len; j++) {
54                    current_dist = this.get_distance(this.arr[i],
                          this.arr[j]);
55                    if (current_dist < min_dist)
56                        min_dist = current_dist;
57                }
58
59            return min_dist;
60        }
61
62        // Получение точек, находящихся от заданной точки на расстоянии,
63        // не превышающем заданную константу
64        get_points_less(point, max_len) {
65            return this.arr.filter(elem =>
66                this.get_distance(point, elem) <= max_len);
67        }
68
69        // axis: 0-x, 1-y;
70        // direction: 0-'+', 1-'-';
71        get_points_axis(axis, direction) {
```

```
            let func;

            if (!axis && !direction)
                func = p => p.x > 0;
            else if (!axis && direction)
                func = p => p.x < 0;
            else if (!direction)
                func = p => p.y > 0;
            else
                func = p => p.y < 0;

            return this.arr.filter(func);
        }
        get_points_inside_rectangle(min_x, max_x, min_y, max_y) {
            return this.arr.filter(p =>
                p.x > min_x && p.x < max_x &&
                p.y > min_y && p.y < max_y);
        }
    }
```

Листинг 6 — Код тестов. Задание 3.

```
    function test_points() {
        let points = new Points();
        points.add("p", 0, 0);
        points.add("p0", 3, 4);
        points.add("p1", 1, 1);
        points.add("p2", 10, 10);
        points.add("p3", 1, 10);
        points.add("p4", 10, 1);
        points.add("p5", 12, 0);
        points.add("p6", -12, 1);
        points.add("p7", 12, -1);

        points.log();

        console.log(points.read("p1"));
        console.log(points.read("p3"));
        console.log(points.read("p13"));
        console.log(points.read("p5"));

        points.update("p5", 100, 12);
        points.log();
        points.delete("p5")
        points.log();

        console.log(points.min_distance());
```

```
26
27          let p = points.read("p"); // 0, 0
28          console.log(points.get_points_less(p, 5));
29
30          console.log("Points:\n")
31          points.log();
32          console.log("axis X-:\n", points.get_points_axis(0, 1));
33          console.log("axis Y-:\n", points.get_points_axis(1, 1));
34
35          console.log("rectangle: -10, 10, -10, 10\n",
                points.get_points_inside_rectangle(-10, 10, -10, 10));
36      }
```

TASK_2. Задание 1

Техническое задание:

— Создать класс Точка.

— Добавить классу точка Точка метод инициализации полей и метод вывода полей на экран

— Создать класс Отрезок.

— У класса Отрезок должны быть поля, являющиеся экземплярами класса Точка.

— Добавить классу Отрезок метод инициализации полей, метод вывода информации о полях на экран, а так же метод получения длины отрезка.

Листинг 7 — Код программы. Задание 1.

```
1     class Point {
2         constructor(x, y) {
3             this.set_data(x, y);
4         }
5
6         set_data(x, y) {
7             this.x = x;
8             this.y = y;
9         }
10
11        log() {
12            console.log("X: ", this.x, "Y: ", this.y);
13        }
14    }
15
16    class Line {
17        constructor(x1, y1, x2, y2) {
18            this.set_data(x1, y1, x2, y2);
19        }
20
21        set_data(x1, y1, x2, y2) {
22            this.start_point = new Point(x1, y1);
23            this.end_point = new Point(x2, y2);
24        }
25
26        get_distance() {
27            const dx = this.start_point.x - this.end_point.x;
28            const dy = this.start_point.y - this.end_point.y;
29            return Math.sqrt(dx * dx + dy * dy);
30        }
31
```

```
32          log() {
33              console.log("Начальная точка:");
34              this.start_point.log();
35              console.log("Конечная точка:");
36              this.end_point.log();
37          }
38      }
```

TASK_2. Задание 2

Техническое задание:

— Создать класс Треугольник.

— Класс Треугольник должен иметь поля, хранящие длины сторон треугольника.

— Необходимо обеспечить уникальность номеров студенческих билетов.

Реализовать следующие методы:

— Метод инициализации полей

— Метод проверки возможности существования треугольника с такими сторонами

— Метод получения периметра треугольника

— Метод получения площади треугольника

— Метод для проверки факта: является ли треугольник прямоугольным

Листинг 8 — Код программы. Задание 1.

```
class Triangle {
    constructor(a, b, c) {
        this.set_data(a, b, c);
    }

    set_data(a, b, c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    check_existence() {
        if ((this.a < this.b + this.c) &&
            (this.b < this.a + this.c) &&
            (this.c < this.b + this.a))
            return true;
        return false;
    }

    get_perimeter() {
        if (!this.check_existence())
            return;
        return this.a + this.b + this.c;
    }

    area() {
```

```
27              if (!this.check_existence())
28                  return;
29              let semi_perimeter = this.get_perimeter() / 2;
30              return Math.sqrt(semi_perimeter * (semi_perimeter - this.a) *
31                  (semi_perimeter - this.b) * (semi_perimeter - this.c));
32          }
33
34          check_rectangular() {
35              let EPS = 1e-5;
36              if (!this.check_existence())
37                  return;
38              let a = this.a, b = this.b, c = this.c;
39
40              if ((a * a + b * b - c * c < EPS) ||
41                  (a * a + c * c - b * b < EPS) ||
42                  (c * c + b * b - a * a < EPS))
43                  return true;
44              return false;
45          }
46      }
```

TASK_2. Задание 3

Техническое задание:

— Реализовать программу, в которой происходят следующие действия:
— Происходит вывод целых чисел от 1 до 10 с задержками в 2 секунды.
— После этого происходит вывод от 11 до 20 с задержками в 1 секунду.
— Потом опять происходит вывод чисел от 1 до 10 с задержками в 2 секунды.
— После этого происходит вывод от 11 до 20 с задержками в 1 секунду.
— Это должно происходить циклически.

Листинг 9 — Код программы. Задание 3

```
"use strict"

const TIME_FIRST = 1000
const TIME_SECOND = 500

function timer() {
    let a = 1;

    let funcFirst = () => {
        console.log(a++);
        if (a > 10) {
            clearInterval(interval1);
            interval2 = setInterval(funcSecond, TIME_SECOND)
        }
    }

    let funcSecond = () => {
        console.log(a++);
        if (a > 20) {
            clearInterval(interval2);
            a = 1;
            interval1 = setInterval(funcFirst, TIME_FIRST)
        }
    }

    let interval1 = setInterval(funcFirst, TIME_FIRST);
    let interval2;
}
```