«Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Савченко В.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

Лабораторной работе №3

«Первая итерация проекта системы»

Студент группы ПИ 92 В.М. Шульпов

Преподаватель Савченко В.В.

Барнаул 2022

**Задание**

|  |  |  |  |
| --- | --- | --- | --- |
| № | ФИО студента  группы ПИ-92 | Прикладная область | Задание |
| 219 | Шульпов Виктор Максимович | Игровые системы | Проектирование обучающего игрового комплекса для младших школьников |

**Описание классов:**

1. **Класс Player** является классом игрока. Он хранит некоторую информацию об игроке.

* Свойство nickname – игровое имя игрока.
* Свойство gamesStat – статистика игрока (объект класса GamesStat).
* Метод changeNickname(nickname) изменяет имя игрока
* Конструктор, принимающий имя игрока и статистику (имя будет получено в процессе запроса имени от пользователя, а статистика будет взята из заранее заготовленного объекта пустой статистики. В конструкторе создается объект класса GamesStat (композиция).
* Сеттеры и геттеры по всем свойствам.

1. **Класс GamesStat** является классом статистики игрока. Объект данного класса самостоятельным не является и создается вместе с объектом класса Player (жизненный цикл этих объектов совпадает).

* Свойство gamesStat хранит список объектов-статистик каждой игры (в данном случае 3 игры).
* Метод changeGameStat(gameName, experience, exercises) изменяет статистику определенной игры.
* Метод writeStatINFile(filename) записывает статистику по данному игроку в файл.
* Метод readStatFromFile(filename) читает статистику по данному игроку из файла в gamesStat.
* Сеттеры и геттеры по всем свойствам.

1. **Класс Game** является классом игры, от него наследуются конкретные игры. Объекты данного класса (или наследники) создаются после создания игрока.

* Свойство name хранит имя игры (передается внутри gameStat через параметр конструктора).
* Свойство experience (передается внутри gameStat через параметр конструктора) хранит опыт игрока в текущей игре.
* Свойство experienceForOnExercise (передается внутри gameConfig через параметр конструктора) хранит опыт, получаемый за одно решенное «упражнение».
* Свойство answeredExercisesNum (передается внутри gameStat через параметр конструктора) хранит количество отвеченных «упражнений».
* Свойство playerAnswer хранит ответ пользователя на вопрос в «упражнении».
* Свойство correctAnswer хранит правильный ответ на вопрос в «упражнении». *Динамическая типизация javascript позволяет хранить в этом свойстве для разных игр разные типы (например, для математической игры – число, а для игры по английскому языку – строку).*
* Метод displayAnswerResultMessage(isAnswerTrue) выводит результат ответа на вопрос (правильно/неправильно)
* Метод goToNextLevel() переводит игрока на следующее «упражнение»
* Метод addExperience() добавляется опыт в текущей игре исходя из параметра конфига с количеством добавляемого опыта.
* Метод addAnsweredExercises() увеличивает счетчик отвеченных упражнений на 1.
* Метод checkCorrectAnswer() сравнивает ответа пользователя и правильный ответ. Возвращает true, если совпали.
* Сеттеры и геттеры по всем свойствам.

1. **Класс GameConfig** является классом конфигурации игры. Объект данного класса является самостоятельным и создается вне объекта класса Player (жизненный цикл этих объектов может не совпадать).

* Свойство gameConfig хранит конфигурацию данной игры.
* Метод changeGameChange(gameConfig) изменяет конфигурацию данной игры.
* Метод writeConfigInFile(filename) записывает конфигурацию игры в файл.
* Метод readConfigFromFile(filename) читает конфигурацию игры в gameConfig
* Сеттеры и геттеры по всем свойствам.

1. **Класс MathGame** является классом математической игры (наследник класса Game).

* Свойство minNumber хранит минимальное число для загадывания программой (передается внутри gameConfig через параметр конструктора).
* Свойство maxNumber хранит максимальное число для загадывания программой (передается внутри gameConfig через параметр конструктора).
* Свойство curNumber1 хранит случайно сгенерированное число 1 для загадывания программой.
* Свойство curNumber2 хранит случайно сгенерированное число 2 для загадывания программой.
* Метод generateRandomNums() генерирует сразу 2 случайных числа в заданном конфигурацией диапазоне.
* Метод countRandNumsSum() считает сумму этих двух чисел.
* Метод displayAsk выводит вопрос на экран.
* Сеттеры и геттеры по всем свойствам.

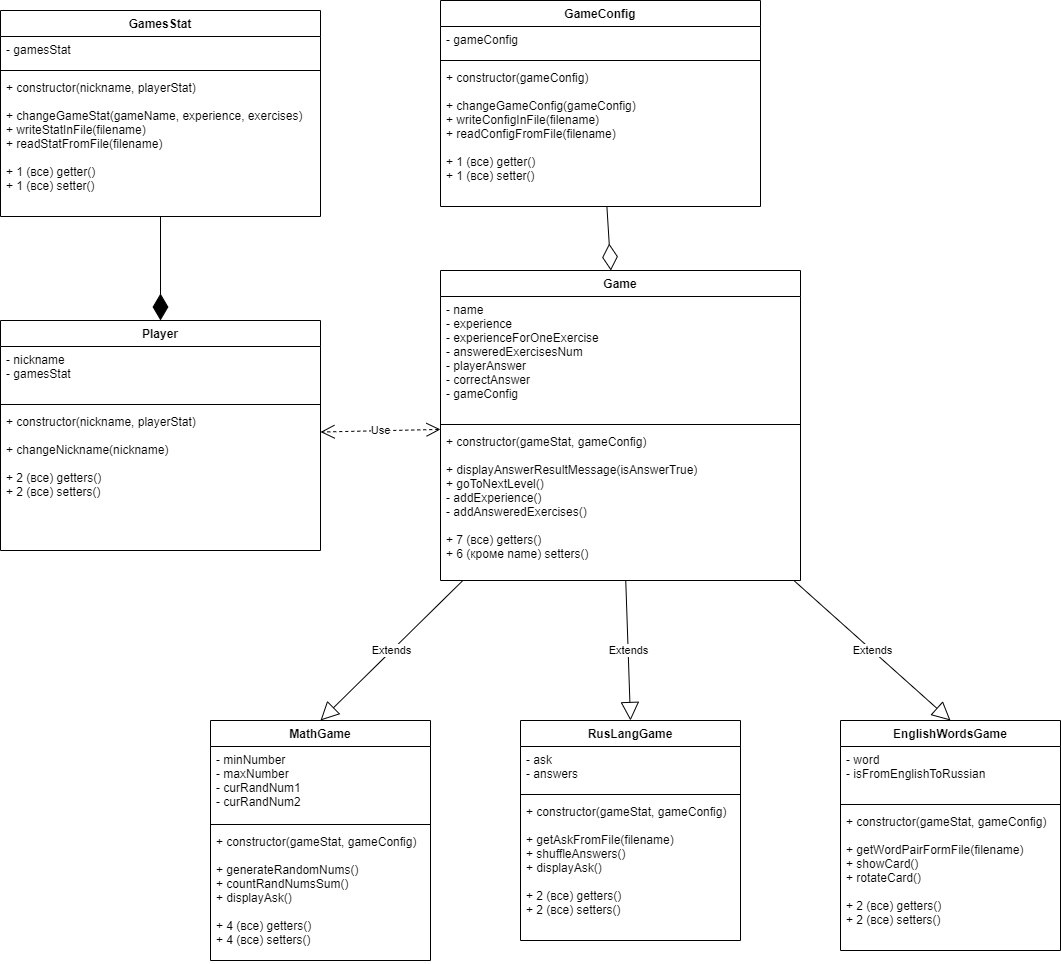
1. **Класс RusLangGame** является классом игры по русскому (наследник класса Game).

* Свойство ask хранит вопрос для «упражнения» (он берется из файла).
* Свойство answers хранит список возможных ответов на вопрос (берется из файла).
* Метод getAskFromFile(filename) читает вопрос и варианты ответа из файла.
* Метод shuffleAnswers() перемешивает элементы списка с вариантами ответа.
* Метод displayAsk() выводит вопрос на экран.
* Сеттеры и геттеры по всем свойствам.

1. **Класс EnglishWordsGame** является классом игры по изучению английских слов (наследник класса Game).

* Свойство word хранит слово из файла (в зависимости от флага хранится русское или английское слово).
* Свойство isFromEnglishToRussian – флаг показывания английского слова и ввода русского при true (при false – наоборот).
* Метод getWordPairFromFile(filename) получает английское слово и его перевод из файла.
* Метод showCard() выводит карточку со словом на экран.
* Метод rotateCard() показывает перевод слова, переворачивая карточку.
* Сеттеры и геттеры по всем свойствам.

**Диаграмма классов**



**P.S.** решил использовать агрегацию, так как в будущем, возможно, будет набор стандартных конфигов (объектов), которые пользователь сможет выбрать себе и заменить одним из них свой. То есть работы с конфигами будут происходить отдельно от объекта игрока. В иной ситуации использование композиции было бы логичнее.

*Однако стоит отметить, что не все поля конфигурации можно изменять обычному пользователю (например, цветовую тему, которая будет в будущей реализации он изменить может, а количество опыта, получаемое за «упражнение» - нет).*

**Возможное расширение системы:**

* Добавление новых игр-наследников.
* Добавление дополнительных полей в конфигурацию. Возможно, касаемые оформления той или иной игры.
* Добавление выбора из стандартных видов конфигураций системы.
* Добавление свойства-флага «премиум аккаунт» для игрока и конфигураций с особенными повышенными параметрами, влияющими на получение опыта.
* Добавление свойства для учета количества ошибок в статистику игрока.

**Начальный код**

const RUS\_LANG\_QUESTIONS\_FILENAME = 'rus\_lang\_questions.json';

const ENGLISH\_WORDS\_TRANSLATION\_FILENAME = 'english\_words.json';

const GAMER\_STAT = 'gamer\_stat.json';

const MATH\_GAME\_NAME = "Математика";

const RUS\_LANG\_GAME\_NAME = "Русский язык";

const ENGLISH\_WORDS\_GAME\_NAME = "Английский язык";

const INIT\_STAT\_ARRAY = {

    gameNumber: 3,

    games: [

        {

            gameName: MATH\_GAME\_NAME,

            experience: 0,

            answeredExercisesNum: 0,

        },

        {

            gameName: RUS\_LANG\_GAME\_NAME,

            experience: 0,

            answeredExercisesNum: 0,

        },

        {

            gameName: ENGLISH\_WORDS\_GAME\_NAME,

            experience: 0,

            answeredExercisesNum: 0,

        }

    ]

}

const MATH\_GAME\_CONFIG = {

    experienceForOneExercise: 100,

    minNumber: -10,

    maxNumber: 10,

}

const RUS\_LANG\_CONFIG = {

    experienceForOneExercise: 100,

}

const ENGLISH\_WORDS\_GAME\_CONFIG = {

    experienceForOneExercise: 50,

}

// получение произвольного случайного числа

function getRandomArbitrary(min, max) {

    return Math.random() \* (max - min) + min;

}

// класс Конфигурации игры

class GameConfig {

    #gameConfig;

    constructor(\_gameConfig) {

        this.#gameConfig = \_gameConfig;

    }

    changeGameConfig(gameConfig) {

        // TODO изменение конфига

    }

    writeConfigInFile(filename) {

        // TODO запись объекта в json файл

    }

    readConfigFromFile(filename) {

        // TODO чтение объекта json файла в переменную

    }

    // getters и setters

    get gameConfig() {

        return this.#gameConfig;

    }

    set gameConfig(value) {

        this.#gameConfig = value;

    }

}

//класс Статистика игрока

class GamesStat {

    #gamesStat;

    constructor(\_gamesStat) {

        this.#gamesStat = [];

        for(let i=0 ; i < \_gamesStat.gameNumber; i++) {

            this.gamesStat.push(\_gamesStat.games[i]);

        }

    }

    changeGameStat(gameName, experience, exercises) {

        gameNum = this.gamesStat.gameNumber;

        for(let i=0; i < gameNum; i++) {

            if (this.gamesStat.games[i].gameName == gameName) {

                this.gamesStat.games[i].experience = experience;

                this.gamesStat.games[i].answeredExercisesNum = exercises;

            }

        }

    }

    writeStatInFile(filename) {

        // TODO запись объекта в json файл

    }

    readStatFromFile(filename) {

        // TODO чтение объекта json файла в переменную

    }

    // getters и setters

    get gamesStat() {

        return this.#gamesStat;

    }

    set gamesStat(value) {

        this.#gamesStat = value;

    }

}

// класс Игрок

class Player {

    #nickname;

    #gamesStat;

    constructor(\_nickname, \_gamesStat) {

        this.#nickname = \_nickname;

        this.#gamesStat = new GamesStat(\_gamesStat);

    }

    changeNickname(\_nickname) {

        this.nickname = \_nickname;

    }

    // getters и setters

    get nickname() {

        return this.#nickname;

    }

    set nickname(value) {

        this.#nickname = value;

    }

    get gamesStat() {

        return this.#gamesStat;

    }

    set gamesStat(value) {

        this.#gamesStat = value;

    }

}

// класс игра

class Game {

    #playerAnswer; //ответ пользователя

    #correctAnswer; //правильный ответ

    #name;

    #experience;

    #experienceForOneExercise;

    #answeredExercisesNum;

    constructor(gameStat, gameConfig) {

        this.#name = gameStat.name;

        this.#experience = gameStat.experience; //инит кол-ва опыта

        this.#experienceForOneExercise = gameConfig.experienceForOneExercise; //инит кол-ва опыта за прохождение 1 упражнения (вопроса)

        this.#answeredExercisesNum = gameStat.addAnsweredExercises; //кол-во отвеченных заданий (вопросов/слов/примеров...)

    }

    displayAnswerResultMessage(isAnswerTrue) {

        if (isAnswerTrue) {

            console.log('Правильный ответ :)');

            // TODO вывод popup в пользовательском интерфейсе

        }

        else {

            console.log('Неправильный ответ :(');

            // TODO вывод popup в пользовательском интерфейсе

        }

    }

    goToNextLevel() {

        console.log('Перешли на следующий уровень!');

        // TODO переход на следующий уровень в интерфейсе

    }

    addExperience() {

        console.log('Добавили опыта!')

        this.experience += this.experienceForOneExercise;

        // TODO добавление опыта текущему игроку

    }

    addAnsweredExercises() {

        console.log('Прошли ещё одно задание!')

        this.answeredExercisesNum += 1;

    }

    checkCorrectAnswer() {

        if (this.playerAnswer == this.correctAnswer) {

            this.addAnsweredExercises();

            this.addExperience();

            return true;

        }

        else return false;

    }

    // getters и setters

    get experience() {

        return this.#experience;

    }

    set experience(value) {

        this.#experience = value;

    }

    get answeredExercisesNum() {

        return this.#answeredExercisesNum;

    }

    set answeredExercisesNum(value) {

        this.#answeredExercisesNum = value;

    }

    get experienceForOneExercise() {

        return this.#experienceForOneExercise;

    }

    set experienceForOneExercise(value) {

        this.#experienceForOneExercise = value;

    }

    get correctAnswer() {

        return this.#correctAnswer;

    }

    set correctAnswer(value) {

        this.#correctAnswer = value;

    }

    get playerAnswer() {

        return this.#playerAnswer;

    }

    set playerAnswer(value) {

        this.#playerAnswer = value;

    }

    // сеттера для имени игры нет

    get name() {

        return this.#name;

    }

}

// класс Математическая игра

class MathGame extends Game {

    #minNumber;

    #maxNumber;

    #curRandNum1;

    #curRandNum2;

    constructor(gameStat, gameConfig) {

        super.constructor(gameStat, gameConfig);

        this.#minNumber = gameConfig.minNumber;

        this.#maxNumber = gameConfig.maxNumber;

        this.#curRandNum1 = 0; //инит случайного числа 1

        this.#curRandNum2 = 0; //инит случайного числа 2

    }

    generateRandomNums() {

        this.curRandNum1 = getRandomArbitrary(this.minNumber, this.maxNumber);

        this.curRandNum2 = getRandomArbitrary(this.minNumber, this.maxNumber);

    }

    countRandNumsSum() {

        this.correctAnswer = this.curRandNum1 + this.curRandNum2

    }

    displayAsk() {

        console.log('Какова сумма чисел (' + str(this.curRandNum1) + ') и (' + str(this.curRandNum2) + ')?');

        // TODO вывод вопроса в интерфейса

    }

    // getters и setters

    get minNumber() {

        return this.#minNumber;

    }

    set minNumber(value) {

        this.#minNumber = value;

    }

    get maxNumber() {

        return this.#maxNumber;

    }

    set maxNumber(value) {

        this.#maxNumber = value;

    }

    get curRandNum1() {

        return this.#curRandNum1;

    }

    set curRandNum1(value) {

        this.#curRandNum1 = value;

    }

    get curRandNum2() {

        return this.#curRandNum2;

    }

    set curRandNum2(value) {

        this.#curRandNum2 = value;

    }

}

// класс Игра на знания русского языка

class RusLangGame extends Game {

    #ask;

    #answers;

    constructor(gameStat, gameConfig) {

        super.constructor(gameStat, gameConfig);

        this.#ask = ''; //инит вопроса

        this.#answers = Array(); //инит списка ответов

    }

    getAskFromFile(filename) {

        // TODO получение вопроса из файла

        question = '';

        return question;

    }

    shuffleAnswers() {

        // TODO перемешать ответы

    }

    displayAsk() {

        console.log('Вопрос: ' + str(this.ask) + ' Варианты ответов: ' + str(this.answers));

        // TODO вывод вопроса в интерфейса

    }

    // getters и setters

    get ask() {

        return this.#ask;

    }

    set ask(value) {

        this.#ask= value;

    }

    get answers() {

        return this.#answers;

    }

    set answers(value) {

        this.#answers = value;

    }

}

// класс Игра на изучение английского языка

class EnglishWordsGame extends Game {

    #word;

    #isFromEnglishToRussian;

    constructor(gameStat, gameConfig) {

        super.constructor(gameStat, gameConfig);

        this.#word = ''; //инит слова

        this.#isFromEnglishToRussian = true; //инит режима: с английского на русский (при false наоборот)

    }

    getWordPairFromFile(filename) {

        // TODO получение слова и его перевода из файла

        this.word = '';

        this.correctAnswer = ''; //правильный ответ (перевод слова)

        question = '';

        return question;

    }

    showCard() {

        console.log('Слово: ' + str(this.word) + '; Перевод на обратной стороне: ' + str(this.wordTranslation));

        // TODO вывод слова в интерфейсе

    }

    rotateCard() {

        // TODO поворот карточки (на обратное стороне перевод)

    }

    // getters и setters

    get word() {

        return this.#word;

    }

    set word(value) {

        this.#word = value;

    }

    get isFromEnglishToRussian() {

        return this.#isFromEnglishToRussian;

    }

    set isFromEnglishToRussian(value) {

        this.#isFromEnglishToRussian = value;

    }

}

**Обязательные изменения в коде:**

* Перенести объекты-константы в файлы, написать функции по их получению.
* Удалить геттеры и сеттеры, которые точно не понадобятся.