«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 №8

Структурные 2 - реализация

(Facade, Information Expert )

Студент группы ПИ 92 В.М. Шульпов

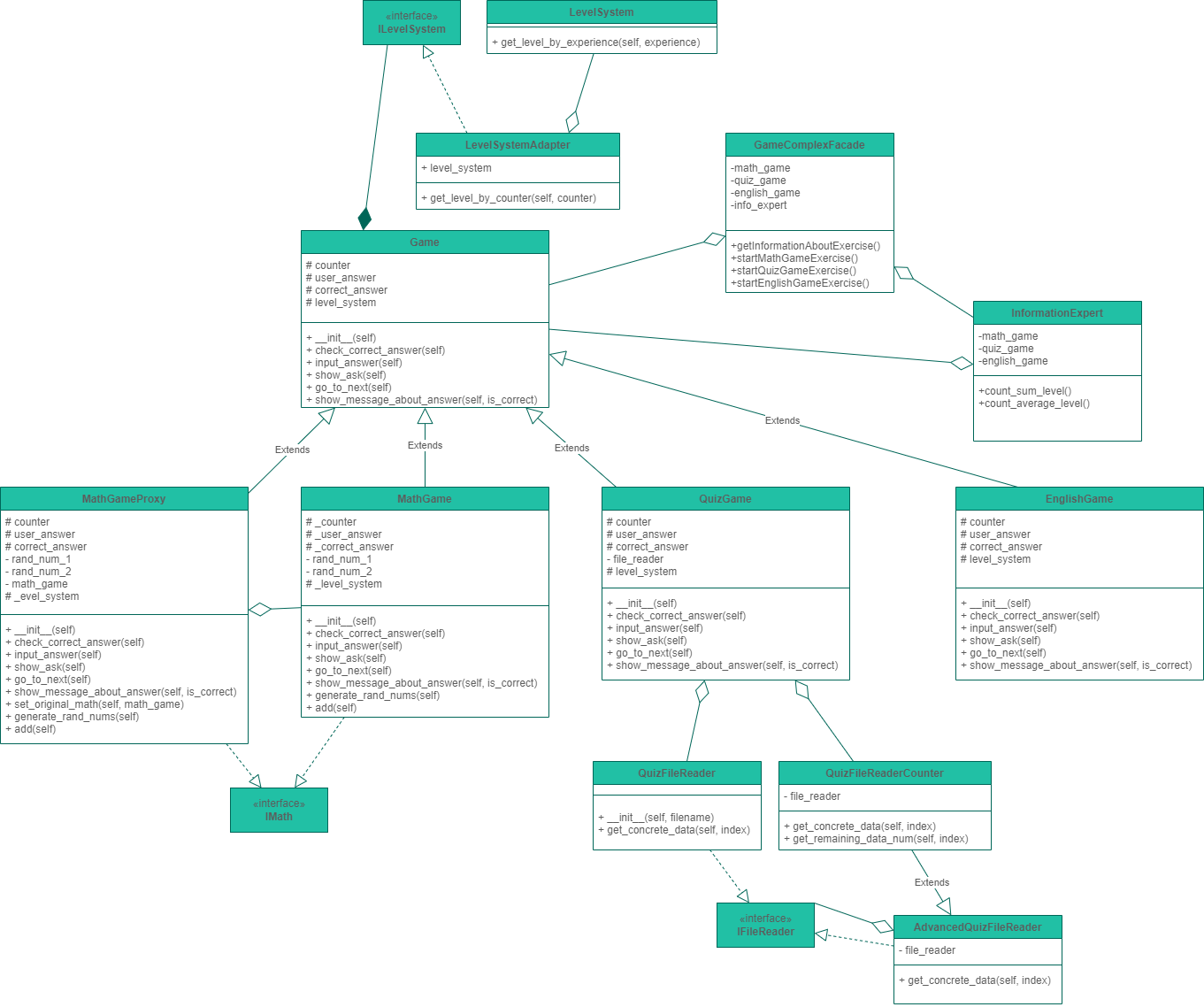
Преподаватель Савченко В.В.

Барнаул 2022

Задание

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

**Диаграмма классов**

****

Краткое описание архитектуры паттерна ФАСАД в программе

Проблема:

Есть сложная система, состоящая из многих взаимодействующих элементов.

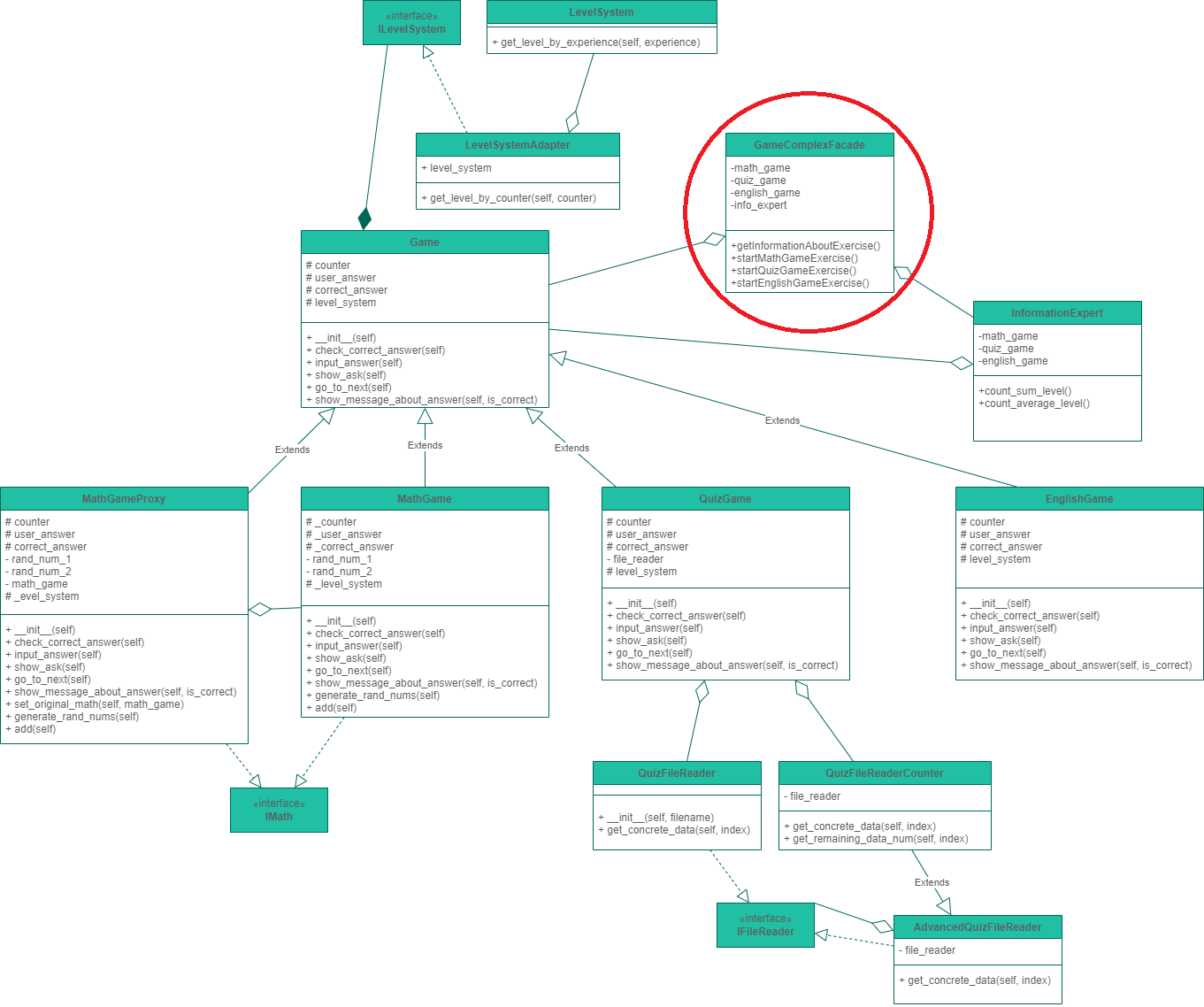
Необходимо упростить процесс работы с ней. Однако система должна остаться открытой для непосредственного использования.

Решение:

Предоставить унифицированный интерфейс для работы с группой интерфейсов

системы.

Диаграмма классов:

~~~~

Выполнение каждой операции фасад делегирует соответствующему компоненту системы.

Код:

class GameComplexFacade:  
 *"""Класс фасад для реализации простого интерфейса работы с играми"""* def \_\_init\_\_(self, info\_expert, math\_game\_proxy, quiz\_game, english\_game):  
 self.info\_expert = info\_expert  
 self.math\_game\_proxy = math\_game\_proxy  
 self.quiz\_game = quiz\_game  
 self.english\_game = english\_game  
  
 def show\_information\_about\_level(self):  
 *"""получить информации, связанную с опытом"""* print(f'Суммарный уровень для всех игр {self.info\_expert.count\_sum\_level()}')  
 print(f'Средний уровень для всех игр {self.info\_expert.count\_average\_level()}')  
  
 def start\_math\_game\_exercise(self):  
 *"""запустить упражнение для математической игры"""* self.math\_game\_proxy.generate\_rand\_nums()  
 self.math\_game\_proxy.add()  
 self.math\_game\_proxy.show\_ask()  
 self.math\_game\_proxy.input\_answer()  
 is\_correct = self.math\_game\_proxy.check\_correct\_answer()  
 self.math\_game\_proxy.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.math\_game\_proxy.go\_to\_next()  
  
 def start\_quiz\_game\_exercise(self):  
 *"""запустить упражнение для игры-викторины"""* if self.quiz\_game.show\_ask():  
 self.quiz\_game.input\_answer()  
 is\_correct = self.quiz\_game.check\_correct\_answer()  
 self.quiz\_game.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.quiz\_game.go\_to\_next()  
  
 def start\_english\_game\_exercise(self):  
 *"""запустить упражнение для игры-обучалки английского языка"""* if self.english\_game.show\_ask():  
 self.english\_game.input\_answer()  
 is\_correct = self.english\_game.check\_correct\_answer()  
 self.english\_game.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.english\_game.go\_to\_next()

**main.py**

def main():  
 print('Обучающий игровой комплекс для младших школьников')  
 quiz\_game = QuizGame(QUIZ\_ASK\_FILE)  
 math\_game\_proxy = MathGameProxy()  
 math\_game = MathGame()  
 math\_game\_proxy.set\_original\_math(math\_game)  
 english\_game = EnglishGame(ENGLISH\_WORDS\_FILE)  
 info\_expert = InformationExpert(math\_game\_proxy, quiz\_game, english\_game)  
 simple\_game\_interface = GameComplexFacade(info\_expert, math\_game\_proxy, quiz\_game, english\_game)  
  
 while True:  
 choice = input(f'\n{"-" \* 100}\nСделайте выбор:\n\t1 - математика'  
 f'\n\t2 - викторина (по русскому языку)'  
 f'\n\t3 - английские слова'  
 f'\n\t4 - показать информацию об опыте в играх'  
 f'\n\n\t0 - выйти\n:')  
 if choice == '1':  
 simple\_game\_interface.start\_math\_game\_exercise()  
 elif choice == '2':  
 simple\_game\_interface.start\_quiz\_game\_exercise()  
 elif choice == '3':  
 simple\_game\_interface.start\_english\_game\_exercise()  
 elif choice == '4':  
 simple\_game\_interface.show\_information\_about\_level()  
 elif choice == '0':  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Краткое описание архитектуры паттерна ИНФОРМАЦИОННЫЙ ЭКСПЕРТ в программе

Проблема:

В системе должна рассчитываться и т. п. необходимая информация.

Решение:

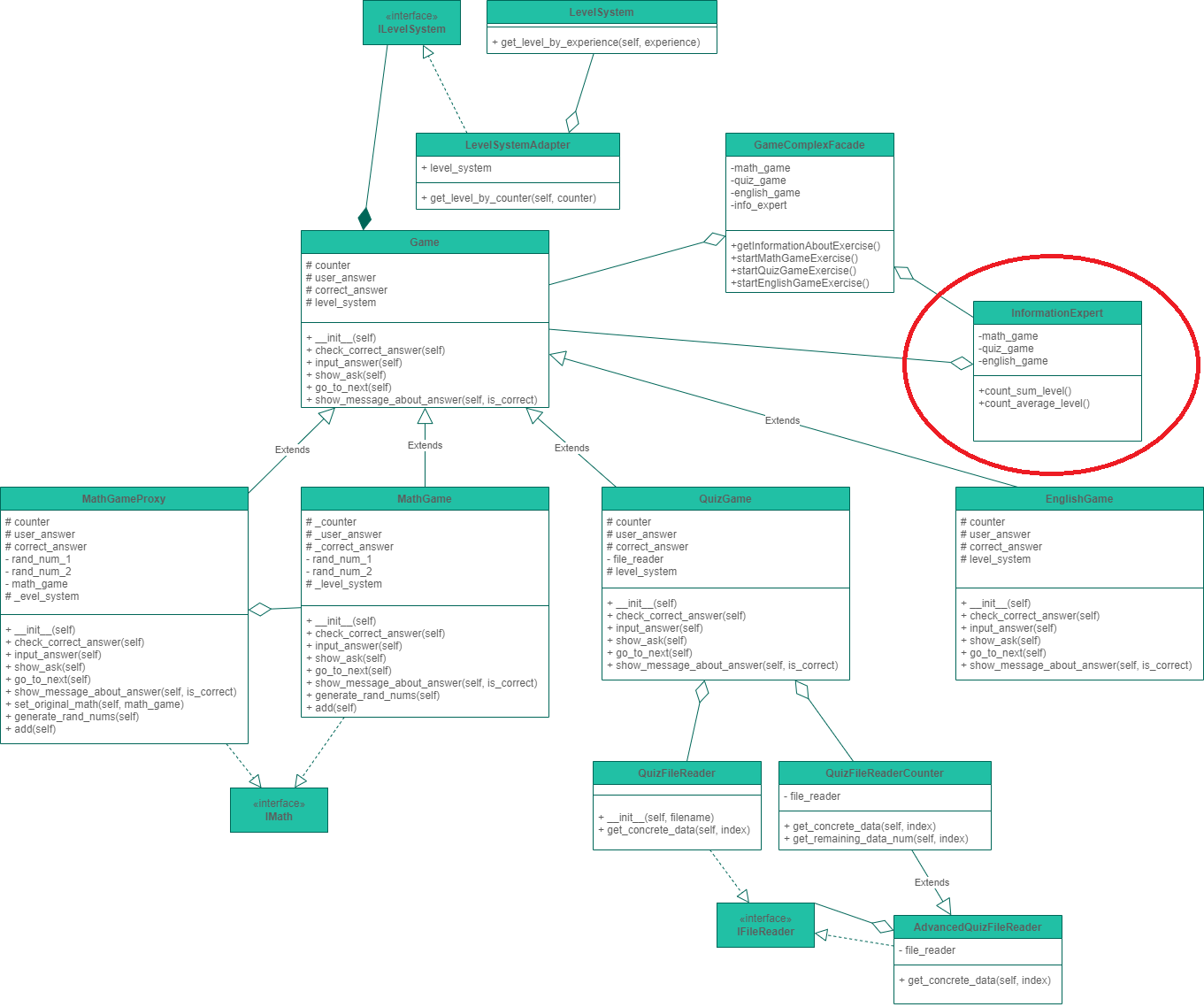
Назначить обязанность аккумуляции информации, расчета и т. п. классу

«информационному эксперту», обладающему необходимой информацией.

Результат:

Преимущество: поддерживает инкапсуляцию, то есть объекты используют свои собственные данные для выполнения поставленных задач.

Диаграмма классов:



Код:

class InformationExpert:  
 *"""информационный эксперт для инкапсулированных в него игр, выполняет расчеты, связанные со свойствами игр"""* def \_\_init\_\_(self, math\_game\_proxy, quiz\_game, english\_game):  
 self.math\_game\_proxy = math\_game\_proxy  
 self.quiz\_game = quiz\_game  
 self.english\_game = english\_game  
  
 def count\_sum\_level(self):  
 *"""подсчитать суммарный опыт для всех игр"""* math\_level = self.math\_game\_proxy.get\_level()  
 quiz\_level = self.quiz\_game.get\_level()  
 english\_level = self.english\_game.get\_level()  
 return math\_level + quiz\_level + english\_level  
  
 def count\_average\_level(self):  
 *"""подсчитать суммарный опыт для всех игр"""* game\_number = 3  
 math\_level = self.math\_game\_proxy.get\_level()  
 quiz\_level = self.quiz\_game.get\_level()  
 english\_level = self.english\_game.get\_level()  
 return (math\_level + quiz\_level + english\_level) / game\_number

**main.py**

from quiz\_game import QuizGame  
from math\_game import MathGame  
from math\_game\_proxy import MathGameProxy  
from english\_game import EnglishGame  
from game\_complex\_facade import GameComplexFacade  
from information\_expert import InformationExpert  
  
QUIZ\_ASK\_FILE = 'quiz\_asks.csv'  
ENGLISH\_WORDS\_FILE = 'english\_words.csv'  
  
  
def main():  
 print('Обучающий игровой комплекс для младших школьников')  
 quiz\_game = QuizGame(QUIZ\_ASK\_FILE)  
 math\_game\_proxy = MathGameProxy()  
 math\_game = MathGame()  
 math\_game\_proxy.set\_original\_math(math\_game)  
 english\_game = EnglishGame(ENGLISH\_WORDS\_FILE)  
 info\_expert = InformationExpert(math\_game\_proxy, quiz\_game, english\_game)  
 simple\_game\_interface = GameComplexFacade(info\_expert, math\_game\_proxy, quiz\_game, english\_game)  
  
 while True:  
 choice = input(f'\n{"-" \* 100}\nСделайте выбор:\n\t1 - математика'  
 f'\n\t2 - викторина (по русскому языку)'  
 f'\n\t3 - английские слова'  
 f'\n\t4 - показать информацию об опыте в играх'  
 f'\n\n\t0 - выйти\n:')  
 if choice == '1':  
 simple\_game\_interface.start\_math\_game\_exercise()  
 elif choice == '2':  
 simple\_game\_interface.start\_quiz\_game\_exercise()  
 elif choice == '3':  
 simple\_game\_interface.start\_english\_game\_exercise()  
 elif choice == '4':  
 simple\_game\_interface.show\_information\_about\_level()  
 elif choice == '0':  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Код программы:

game.py

from level\_system\_adapter import LevelSystemAdapter  
from level\_system import LevelSystem  
  
  
class Game:  
 *"""игра"""* def \_\_init\_\_(self):  
 self.counter = 0  
 self.user\_answer = None  
 self.correct\_answer = None  
 self.level\_system = LevelSystemAdapter(LevelSystem())  
  
 def check\_correct\_answer(self):  
 *"""проверка правильности ответа пользователя"""* if self.user\_answer == self.correct\_answer:  
 return True  
 return False  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* pass  
  
 def show\_ask(self):  
 *"""показ вопроса"""* pass  
  
 def go\_to\_next(self):  
 *"""увеличение счетчика (переход на следующий уровень/вопрос)"""* self.counter += 1  
 print(f'Переход на следующее задание!\nВаш текущий уровень {self.get\_level()}!')  
  
 def show\_message\_about\_answer(self, is\_correct):  
 *"""сообщение о правильности ответа"""* if is\_correct:  
 print(f'Вы ответили правильно! (ответ: {self.correct\_answer})')  
 else:  
 print('Вы ответили неверно!')  
  
 def get\_level(self):  
 *"""получить текущий уровень"""* return self.level\_system.get\_level\_by\_counter(self.counter)

**ifile\_reader.py:**

from abc import ABC, abstractmethod  
  
  
class IFileReader(ABC):  
 *"""читатель файла"""* @abstractmethod  
 def get\_concrete\_data(self, index):  
 *"""получение данных для уровня"""* pass

**ilevel\_system.py:**

from abc import ABC, abstractmethod  
  
class ILevelSystem(ABC):  
 *"""интерфейс уровневой системы"""* @abstractmethod  
 def get\_level\_by\_counter(self, counter):  
 *"""посчитать текущий уровень по количеству пройденных уровней"""* pass

**level\_system.py:**

class LevelSystem():  
 *"""уровневая система"""* def \_\_init\_\_(self):  
 pass  
  
 def get\_level\_by\_experience(self, experience):  
 *"""посчитать текущий уровень по количеству опыта  
 100 опыта = 1 уровень"""* return experience // 100

**level\_system\_adapter.py**

from ilevel\_system import ILevelSystem  
  
  
class LevelSystemAdapter(ILevelSystem):  
 *"""адаптер уровневой системы"""* def \_\_init\_\_(self, level\_system):  
 self.level\_system = level\_system  
  
 def get\_level\_by\_counter(self, counter):  
 *"""посчитать текущий уровень по количеству пройденных уровней  
 1 пройденное задание = 50 опыта  
 100 опыта = 1 уровень"""* experience = 50 \* counter  
 return self.level\_system.get\_level\_by\_experience(experience)

imath.py:

from abc import ABC, abstractmethod  
  
  
class IMath(ABC):  
 *"""интерфейс игры математика (сложение чисел)"""* @abstractmethod  
 def add(self):  
 *"""суммирование 2 чисел"""* pass

math\_game.py:

from game import Game  
from random import randint  
from imath import IMath  
  
  
MIN\_NUM = 1  
MAX\_NUM = 20  
  
  
class MathGame(Game, IMath):  
 *"""игра - суммирование чисел (математика)"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.rand\_num\_1 = None  
 self.rand\_num\_2 = None  
  
 def show\_ask(self):  
 *"""показ вопроса"""* print(f'Вопрос №{self.counter + 1}\n\tЧему равно {self.rand\_num\_1} + {self.rand\_num\_2}?')  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* num = input('Введите сумму чисел:\t')  
 self.user\_answer = num  
  
 def generate\_rand\_nums(self):  
 *"""генерация двух случайных чисел для суммирования"""* self.rand\_num\_1 = randint(MIN\_NUM, MAX\_NUM)  
 self.rand\_num\_2 = randint(MIN\_NUM, MAX\_NUM)  
  
 def add(self):  
 *"""суммирование 2 чисел"""* self.correct\_answer = str(self.rand\_num\_1 + self.rand\_num\_2)  
 return self.correct\_answer

math\_game\_proxy.py:

from game import Game  
from imath import IMath  
from datetime import datetime  
  
  
MATH\_LOG\_FILE = 'math\_log.txt'  
  
  
MIN\_NUM = 1  
MAX\_NUM = 20  
  
  
def log\_add(filename):  
 *"""логирование (хранение истории обращений)"""* with open(filename, 'a') as file:  
 cur\_date = datetime.now()  
 adding\_log = f'addding; время: {cur\_date}\n'  
 file.write(adding\_log)  
  
  
class MathGameProxy(Game, IMath):  
 *"""логирующий заместитель игры - суммирование чисел (математика)"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.math\_game = None  
  
 def set\_original\_math(self, math\_game):  
 *"""ленивая инициализация реального объекта"""* if self.math\_game is None:  
 self.math\_game = math\_game  
  
 def generate\_rand\_nums(self):  
 *"""генерация двух случайных чисел для суммирования"""* self.math\_game.generate\_rand\_nums()  
  
 def add(self):  
 *"""суммирование 2 чисел"""* self.math\_game.add()  
 log\_add(MATH\_LOG\_FILE)  
  
 def show\_ask(self):  
 *"""показать вопрос"""* self.math\_game.show\_ask()  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* self.math\_game.input\_answer()  
  
 def check\_correct\_answer(self):  
 *"""проверка правильности ответа пользователя"""* return self.math\_game.check\_correct\_answer()  
  
 def show\_message\_about\_answer(self, is\_correct):  
 *"""сообщение о правильности ответа"""* self.math\_game.show\_message\_about\_answer(is\_correct)  
  
 def go\_to\_next(self):  
 *"""увеличение счетчика (переход на следующий уровень/вопрос)"""* self.math\_game.go\_to\_next()  
  
 def get\_level(self):  
 *"""получить текущий уровень"""* return self.math\_game.level\_system.get\_level\_by\_counter(self.math\_game.counter)

**advanced\_quiz\_file\_reader.py:**

from ifile\_reader import IFileReader  
import csv  
  
  
class AdvancedQuizFileReader(IFileReader):  
 *"""читатель-декоратор файла для игры викторина  
 (читает вопрос, варианты ответа, и номер правильного ответа)  
 дополняет функционал за счет способности подсчитывать количество оставшихся в файле уровней"""* def \_\_init\_\_(self, file\_reader):  
 pass  
  
 def get\_concrete\_data(self, index):  
 *"""получать вопрос, номер правильного ответа и варианты ответа"""* pass  
  
 def get\_filename(self):  
 *"""getter filename"""* pass  
  
 def \_\_get\_remaining\_data\_num(self, index):  
 *"""подсчитывает количество оставшихся вопросов в файле"""* pass

**quiz\_file\_reader.py:**

from ifile\_reader import IFileReader  
import csv  
  
  
class QuizFileReader(IFileReader):  
 *"""читатель файла для игры викторина (читает вопрос, варианты ответа, и номер правильного ответа"""* def \_\_init\_\_(self, filename):  
 self.filename = filename  
  
 def get\_filename(self):  
 *"""getter filename"""* return self.filename  
  
 def get\_concrete\_data(self, index):  
 *"""получать вопрос, номер правильного ответа и варианты ответа"""* with open(self.filename, 'r', newline='') as File:  
 reader = csv.reader(File, delimiter=',')  
 counter = 0  
 for ask\_i, row in enumerate(reader):  
 ask = row[0]  
 correct\_answer\_num = next(reader)[0]  
 answers = next(reader)  
 counter += 1  
 if ask\_i == index:  
 return [ask, correct\_answer\_num, answers]  
 return None

quiz\_file\_reader\_counter.py

from advanced\_quiz\_file\_reader import AdvancedQuizFileReader  
import csv  
  
  
class QuizFileReaderCounter(AdvancedQuizFileReader):  
 *"""читатель-декоратор файла для игры викторина  
 (читает вопрос, варианты ответа, и номер правильного ответа)  
 дополняет функционал за счет способности подсчитывать количество оставшихся в файле уровней"""* def \_\_init\_\_(self, file\_reader):  
 self.file\_reader = file\_reader  
  
 def get\_concrete\_data(self, index):  
 *"""получать вопрос, номер правильного ответа и варианты ответа"""* counter = self.\_\_get\_remaining\_data\_num(index)  
 print(f'Осталось вопросов: {counter}')  
 return self.file\_reader.get\_concrete\_data(index)  
  
 def get\_filename(self):  
 *"""getter filename"""* return self.file\_reader.get\_filename()  
  
 def \_\_get\_remaining\_data\_num(self, index):  
 *"""подсчитывает количество оставшихся вопросов в файле"""* with open(self.file\_reader.get\_filename(), 'r', newline='') as File:  
 reader = csv.reader(File, delimiter=',')  
 counter = 0  
 for ask\_i, row in enumerate(reader):  
 next(reader)  
 next(reader)  
 if ask\_i < index:  
 continue  
 counter += 1  
 return counter

english\_game.py

from game import Game  
import csv  
  
  
ENGLISH\_WORDS\_FILE = 'english\_words.csv'  
  
  
class EnglishGame(Game):  
 *"""игра - обучалка слов по английскому языку"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.filename = ENGLISH\_WORDS\_FILE  
  
 def \_\_init\_\_(self, filename):  
 super().\_\_init\_\_()  
 self.filename = filename  
  
 def get\_word\_and\_translation(self, index):  
 *"""получение слова и его перевода по индексу (номеру упражнения)"""* with open(self.filename, 'r', newline='') as File:  
 reader = csv.reader(File, delimiter=';')  
 counter = 0  
 for row\_i, row in enumerate(reader):  
 english\_word = row[0]  
 russian\_word = row[1]  
 counter += 1  
 if row\_i == index:  
 return [english\_word, russian\_word]  
 return None  
  
 def show\_ask(self):  
 *"""показ слова """* words = self.get\_word\_and\_translation(self.counter)  
 if words is None:  
 print('Слова кончились!')  
 return False  
 english\_word = words[0]  
 self.correct\_answer = words[1]  
 print(f'Вопрос №{self.counter + 1}\n\tПереведите слово "{english\_word}" на русский язык')  
 return True  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* word = input('Введите перевод:\t')  
 self.user\_answer = word  
  
 def check\_correct\_answer(self):  
 *"""проверка правильности ответа и запись в список неправильных, если ответ неверный"""* if super().check\_correct\_answer():  
 return True  
 return False

quiz\_game.py:

from game import Game  
from quiz\_file\_reader import QuizFileReader  
from quiz\_file\_reader\_counter import QuizFileReaderCounter  
  
QUIZ\_ASK\_FILE = 'quiz\_asks.csv'  
  
  
class QuizGame(Game):  
 *"""игра - викторина (по русскому языку)"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.file\_reader = QuizFileReaderCounter(QuizFileReader(QUIZ\_ASK\_FILE)) *# QuizFileReader(QUIZ\_ASK\_FILE)* def \_\_init\_\_(self, filename):  
 super().\_\_init\_\_()  
 self.file\_reader = QuizFileReaderCounter(QuizFileReader(filename)) *# QuizFileReader(filename)* def show\_ask(self):  
 *"""показ вопроса и вариантов овтета текущего уровня  
 concrete\_data= [ask, num\_correct\_answer, answers]"""* conrete\_data = self.file\_reader.get\_concrete\_data(self.counter)  
 if conrete\_data is None:  
 print('Вопросы кончились!')  
 return False  
 ask = conrete\_data[0]  
 self.correct\_answer = conrete\_data[1]  
 answers = conrete\_data[2]  
 print(f'Вопрос №{self.counter + 1}\n\t{ask}')  
 for i, answer in enumerate(answers):  
 print(f'\t\t{i + 1}) {answer}')  
 return True  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* num = input('Введите номер правильного ответа:\t')  
 self.user\_answer = num

quiz\_asks.csv:

Какая часть слова является носителем его значения? 2

приставка,корень,суффикс,окончание

По какому принципу слова объединяются в части речи? 2

по назначению,по общности грамматических свойств,по смыслу,по историческому принципу

Укажите падеж выделенного имени существительного. Под ДЕРЕВЬЯМИ зеленеют обросшие брусничником кочки. (Соколов-Микитов И.)

4

родительный,дательный,винительный,творительный

Какой род прилагательных не выделяют в русском языке? 4

мужской,средний,женский,общий

Определите род прилагательного, которое употреблено в предложении: «Солнце скрылось за темной тучей».

3

мужской,средний,женский,общий

В каком из словосочетаний употреблено прилагательное мужского рода? 1

летний дождь,длинная дорога,новое платье,сладкая ягода

math\_log.txt:

addding; время: 2022-04-20 02:06:28.384450

addding; время: 2022-04-20 02:06:31.581248

addding; время: 2022-04-20 02:06:46.565661

addding; время: 2022-04-20 02:06:49.972286

addding; время: 2022-04-20 02:06:53.222613

addding; время: 2022-04-20 02:06:55.897448

addding; время: 2022-04-20 02:06:58.671007

addding; время: 2022-04-20 02:06:59.645661

addding; время: 2022-04-20 14:21:02.534213

game\_complex\_facade.py:

class GameComplexFacade:  
 *"""Класс фасад для реализации простого интерфейса работы с играми"""* def \_\_init\_\_(self, info\_expert, math\_game\_proxy, quiz\_game, english\_game):  
 self.info\_expert = info\_expert  
 self.math\_game\_proxy = math\_game\_proxy  
 self.quiz\_game = quiz\_game  
 self.english\_game = english\_game  
  
 def show\_information\_about\_level(self):  
 *"""получить информации, связанную с опытом"""* print(f'Суммарный уровень для всех игр {self.info\_expert.count\_sum\_level()}')  
 print(f'Средний уровень для всех игр {self.info\_expert.count\_average\_level()}')  
  
 def start\_math\_game\_exercise(self):  
 *"""запустить упражнение для математической игры"""* self.math\_game\_proxy.generate\_rand\_nums()  
 self.math\_game\_proxy.add()  
 self.math\_game\_proxy.show\_ask()  
 self.math\_game\_proxy.input\_answer()  
 is\_correct = self.math\_game\_proxy.check\_correct\_answer()  
 self.math\_game\_proxy.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.math\_game\_proxy.go\_to\_next()  
  
 def start\_quiz\_game\_exercise(self):  
 *"""запустить упражнение для игры-викторины"""* if self.quiz\_game.show\_ask():  
 self.quiz\_game.input\_answer()  
 is\_correct = self.quiz\_game.check\_correct\_answer()  
 self.quiz\_game.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.quiz\_game.go\_to\_next()  
  
 def start\_english\_game\_exercise(self):  
 *"""запустить упражнение для игры-обучалки английского языка"""* if self.english\_game.show\_ask():  
 self.english\_game.input\_answer()  
 is\_correct = self.english\_game.check\_correct\_answer()  
 self.english\_game.show\_message\_about\_answer(is\_correct)  
 if is\_correct:  
 self.english\_game.go\_to\_next()

information\_expert.py:

from game import Game

class InformationExpert:  
 *"""информационный эксперт для инкапсулированных в него игр, выполняет расчеты, связанные со свойствами игр"""* def \_\_init\_\_(self, math\_game\_proxy, quiz\_game, english\_game):  
 self.math\_game\_proxy = math\_game\_proxy  
 self.quiz\_game = quiz\_game  
 self.english\_game = english\_game  
  
 def count\_sum\_level(self):  
 *"""подсчитать суммарный опыт для всех игр"""* math\_level = self.math\_game\_proxy.get\_level()  
 quiz\_level = self.quiz\_game.get\_level()  
 english\_level = self.english\_game.get\_level()  
 return math\_level + quiz\_level + english\_level  
  
 def count\_average\_level(self):  
 *"""подсчитать суммарный опыт для всех игр"""* game\_number = 3  
 math\_level = self.math\_game\_proxy.get\_level()  
 quiz\_level = self.quiz\_game.get\_level()  
 english\_level = self.english\_game.get\_level()  
 return (math\_level + quiz\_level + english\_level) / game\_number

quiz\_game.py:

from game import Game  
from quiz\_file\_reader import QuizFileReader  
from quiz\_file\_reader\_counter import QuizFileReaderCounter  
  
QUIZ\_ASK\_FILE = 'quiz\_asks.csv'  
  
  
class QuizGame(Game):  
 *"""игра - викторина (по русскому языку)"""* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.file\_reader = QuizFileReaderCounter(QuizFileReader(QUIZ\_ASK\_FILE)) *# QuizFileReader(QUIZ\_ASK\_FILE)* def \_\_init\_\_(self, filename):  
 super().\_\_init\_\_()  
 self.file\_reader = QuizFileReaderCounter(QuizFileReader(filename)) *# QuizFileReader(filename)* def show\_ask(self):  
 *"""показ вопроса и вариантов овтета текущего уровня  
 concrete\_data= [ask, num\_correct\_answer, answers]"""* conrete\_data = self.file\_reader.get\_concrete\_data(self.counter)  
 if conrete\_data is None:  
 print('Вопросы кончились!')  
 return False  
 ask = conrete\_data[0]  
 self.correct\_answer = conrete\_data[1]  
 answers = conrete\_data[2]  
 print(f'Вопрос №{self.counter + 1}\n\t{ask}')  
 for i, answer in enumerate(answers):  
 print(f'\t\t{i + 1}) {answer}')  
 return True  
  
 def input\_answer(self):  
 *"""ввод пользовательского ответа"""* num = input('Введите номер правильного ответа:\t')  
 self.user\_answer = num

main.py

from quiz\_game import QuizGame  
from math\_game import MathGame  
from math\_game\_proxy import MathGameProxy  
from english\_game import EnglishGame  
from game\_complex\_facade import GameComplexFacade  
from information\_expert import InformationExpert  
  
QUIZ\_ASK\_FILE = 'quiz\_asks.csv'  
ENGLISH\_WORDS\_FILE = 'english\_words.csv'  
  
  
def main():  
 print('Обучающий игровой комплекс для младших школьников')  
 quiz\_game = QuizGame(QUIZ\_ASK\_FILE)  
 math\_game\_proxy = MathGameProxy()  
 math\_game = MathGame()  
 math\_game\_proxy.set\_original\_math(math\_game)  
 english\_game = EnglishGame(ENGLISH\_WORDS\_FILE)  
 info\_expert = InformationExpert(math\_game\_proxy, quiz\_game, english\_game)  
 simple\_game\_interface = GameComplexFacade(info\_expert, math\_game\_proxy, quiz\_game, english\_game)  
  
 while True:  
 choice = input(f'\n{"-" \* 100}\nСделайте выбор:\n\t1 - математика'  
 f'\n\t2 - викторина (по русскому языку)'  
 f'\n\t3 - английские слова'  
 f'\n\t4 - показать информацию об опыте в играх'  
 f'\n\n\t0 - выйти\n:')  
 if choice == '1':  
 simple\_game\_interface.start\_math\_game\_exercise()  
 elif choice == '2':  
 simple\_game\_interface.start\_quiz\_game\_exercise()  
 elif choice == '3':  
 simple\_game\_interface.start\_english\_game\_exercise()  
 elif choice == '4':  
 simple\_game\_interface.show\_information\_about\_level()  
 elif choice == '0':  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()