«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 №5

«Структурные паттерны 1 - проектирование

(Adapter, Decorator)»

Студент группы ПИ 92 В.М. Шульпов

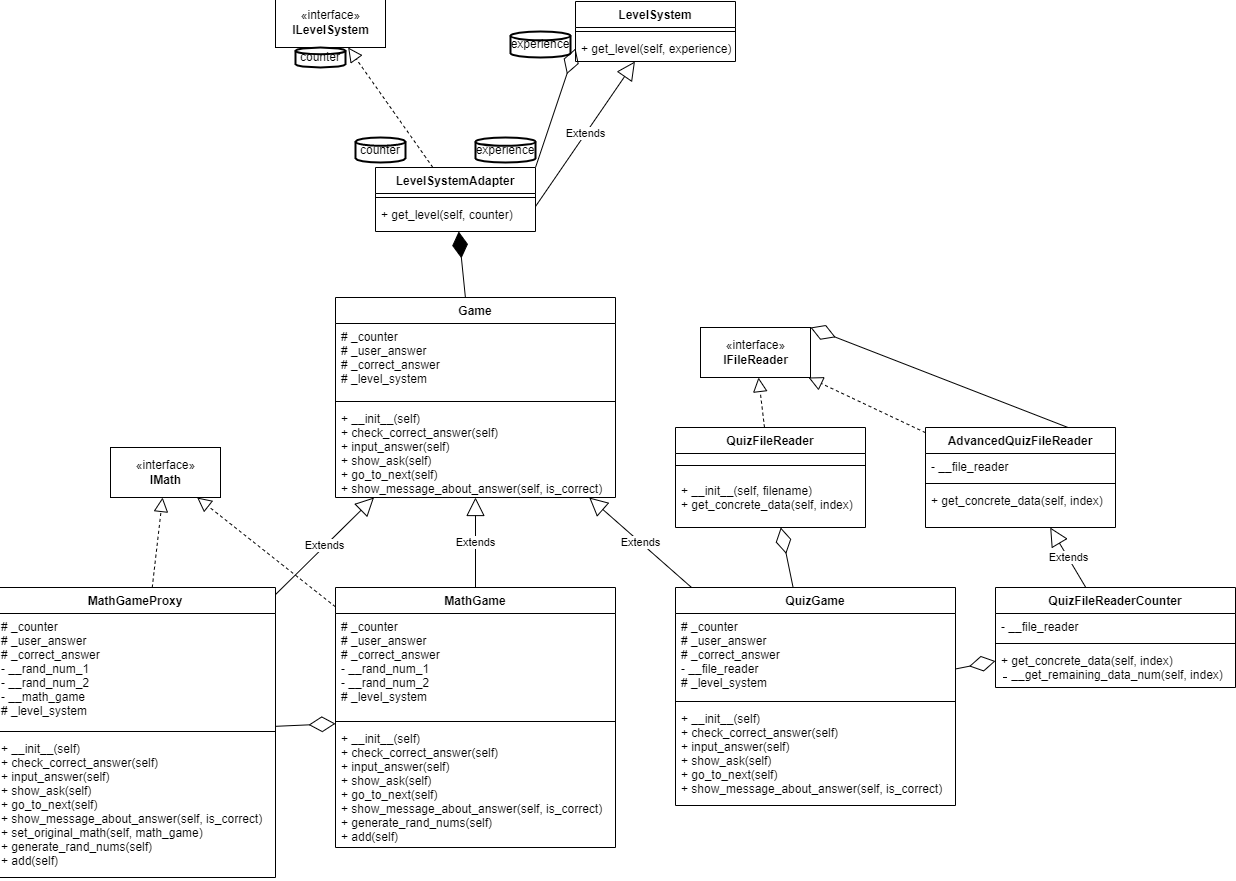
Преподаватель Савченко В.В.

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**

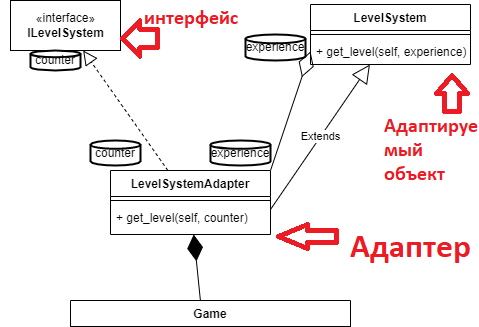


**Краткое описание архитектуры паттерна АДАПТЕР в программе**

Проблема: необходимо обеспечить взаимодействие несовместимых компонентов с разными интерфейсами. У нас есть класс LevelSystem, в котором реализуется получение значения текущего уровня с параметром experience (опыт). Однако клиентский интерфейс ничего не знает про experience, но знает про counter (счетчик выполненных заданий).

Решение: конвертировать исходный интерфейс компонента к другому виду с помощью промежуточного объекта – адаптера. Класс Адаптер реализует интерфейс, известный его клиентам.

Диаграмма классов:



* Адаптер (LevelSystemAdapter) реализует интерфейс ILevelSystem, на который рассчитывает клиент
* Адаптер (LevelSystemAdapter) содержит ссылку на адаптируемый объект (LevelSystem)
* Адаптер реализует все методы интерфейса

Код интерфейса (ILevelSystem):

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

Код адаптера (LevelSystemAdapter):

**from** level\_system **import** LevelSystem  
**from** ilevel\_system **import** ILevelSystem  
  
**class** LevelSystemAdapter(ILevelSystem, LevelSystem):  
 *"""адаптер уровневой системы"""* **def** \_\_init\_\_(self, level\_system):  
 self.level\_system = level\_system  
  
 **def** get\_level(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(experience)

Код адаптируемого объекта (LevelSystem):

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

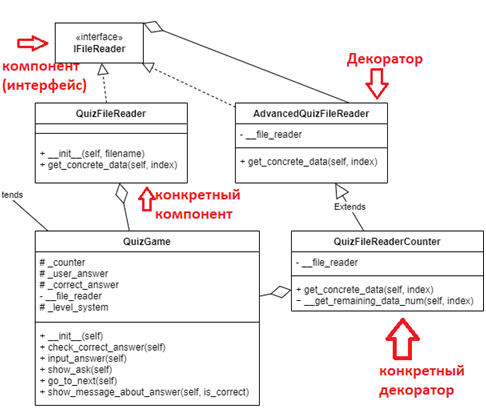
**Краткое описание архитектуры паттерна ДЕКОРАТОР в программе**

Проблема: необходимо возложить дополнительные обязанности на  
отдельный объект, а не на класс в целом. Надо имитировать интерфейс у объекта, который  
таким интерфейсом не обладает.

Решение: динамически добавить объекту новые обязанности, не прибегая при этом к  
порождению подклассов.

Результат: большая гибкость, чем у статического наследования: можно добавлять и удалять обязанности во время выполнения программы, в то время как при использовании наследования надо было бы создавать новый класс для каждой дополнительной обязанности.  
Данный паттерн позволяет избежать перегруженных методами классов на верхних уровнях  
иерархии — новые обязанности можно добавлять по мере необходимости

Диаграмма классов:



* Декоратор (AdvancedQuizFileReader) реализует тот же интерфейс, что и конкретный компонент (QuizFileReader).
* Декоратор (AdvancedQuizFileReader) содержит ссылку на декорируемый объект (IFileReader)
* Декоратор (AdvancedQuizFileReader) дополняет функционал декорируемого объекта (IFileReader)

Код интерфейса (IFileReader):

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

Код конкретного компонента (QuizFileReader)

**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**

Код декоратора (AdvancedQuizFileReader):

**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**

Код конкретного декоратора (QuizFileReaderCounter):

**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

**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(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**

**level\_system.py:**

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

**level\_system\_adapter.py**

**from** level\_system **import** LevelSystem  
**from** ilevel\_system **import** ILevelSystem  
  
  
**class** LevelSystemAdapter(ILevelSystem, LevelSystem):  
 *"""адаптер уровневой системы"""* **def** \_\_init\_\_(self, level\_system):  
 self.level\_system = level\_system  
  
 **def** get\_level(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(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()

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

**from** ifile\_reader **import** IFileReader  
**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

**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

**main.py**

**from** quiz\_game **import** QuizGame  
**from** math\_game **import** MathGame  
**from** math\_game\_proxy **import** MathGameProxy  
  
QUIZ\_ASK\_FILE = **'quiz\_asks.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)  
 **while True**:  
 choice = input(**f'\n{"-"** \* 100**}\nСделайте выбор:\n\t1 - математика\n\t2 - викторина (по русскому языку)'  
 f'\n\n\t0 - выйти\n:'**)  
 **if** choice == **'1'**:  
  
 math\_game\_proxy.generate\_rand\_nums()  
 math\_game\_proxy.add()  
 math\_game\_proxy.show\_ask()  
 math\_game\_proxy.input\_answer()  
 is\_correct = math\_game\_proxy.check\_correct\_answer()  
 math\_game\_proxy.show\_message\_about\_answer(is\_correct)  
 **if** is\_correct:  
 math\_game\_proxy.go\_to\_next()  
  
 **elif** choice == **'2'**:  
 **if** quiz\_game.show\_ask():  
 quiz\_game.input\_answer()  
 is\_correct = quiz\_game.check\_correct\_answer()  
 quiz\_game.show\_message\_about\_answer(is\_correct)  
 **if** is\_correct:  
 quiz\_game.go\_to\_next()  
 **elif** choice == **'3'**:  
 **pass  
 elif** choice == **'0'**:  
 **return  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()