МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Объектно-ориентированое программирование»
Тема: Логирование, перегрузка операций

Студент гр. 1381 _____ Исайкин Г. И.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2022

Цель работы.

Реализовать логирование для игры на языке C++.

Задание.

Реализовать класс/набор классов отслеживающих изменения состояний в программе. Отслеживание должно быть 3-х уровней:

- Изменения состояния игрока и поля, а также срабатывание событий
- Состояние игры (игра начата, завершена, сохранена, и.т.д.)
- Отслеживание критических состояний и ошибок (поле инициализировано с отрицательными размерами, игрок попытался перейти на непроходимую клетку, и.т.д.)
- Реализованы классы для вывода информации разных уровней для в консоль и в файл с перегруженным оператором вывода в поток.

Требования:

- Разработан класс/набор классов отслеживающий изменения разных уровней
- Разработаны классы для вывода в консоль и файл с соблюдением идиомы RAII и перегруженным оператором вывода в поток.
- Разработанные классы спроектированы таким образом, чтобы можно было добавить новый формат вывода без изменения старого кода (например, добавить возможность отправки логов по сети)
- Выбор отслеживаемых уровней логирования должен происходить в runtime
- В runtime должен выбираться способ вывода логов (нет логирования, в консоль, в файл, в консоль и файл)

Примечания:

- Отслеживаемые сущности не должны ничего знать о сущностях, которые их логируют
- Уровни логирования должны быть заданными отдельными классами или перечислением
- Разные уровни в логах должны помечаться своим префиксом
- Рекомендуется сделать класс сообщения
- Для отслеживания изменений можно использовать наблюдателя
- Для вывода сообщений можно использовать адаптер, прокси и декоратор

Выполнение работы.

Для выполнения работы сделаны классы Messange, Observer, Log и наследники LogFile и LogConsole.

Класс Message. Класс носит уровень сообщения и сам текст сообщения, который потом будет выводится в консоль и/или в файл.

Класс Log. Это абстрактный класс, который выводит сообщение туда, куда нам нужно и ставит префикс к сообщению. За это отвечает чисто виртуальный метод void out_message(Message messege). Класс также хранит уровни сообщений, которые надо выводить.

Класс LogFile. Это наследник класса Log. Переопределённый метод void out_message(Message messege) проверяет, соответствует ли уровень сообщения нужному, и если да, выводит сообщение в файл.

Класс LogConsole. Это наследник класса Log. Переопределённый метод void out_message(Message messege) проверяет, соответствует ли уровень сообщения нужному, и если да, выводит сообщение в консоль.

Класс Observer. Это класс наблюдает за объектами класса Field и Player, выводит сообщения о их имениях, о начале/завершении игры и о ошибках. Делается это через методы void set_field(Field *f), void

set_player(Player *p), void log_field_changes(Field *new_field), void log_player_changes(Player *new_player), void log_dead_end(), void log_start_game() и void log_finish_game(GameStatus status). Для вывода класс хранит указатели на объекты наследников класса Log.

Другие изменения.

- Добавлены операторы сравнения == и != для Player, Field и Cell.
- Класс FacadePlayer теперь хранит ссылку на объект класса Observer.
- В фасад FacadePlayer добавлены методы GameStatus game_start() и GameStatus game_step(), которые запускают игровой процесс (включая логирование).
- Реализован класс CommandReader с его методами Direction get_direction(), который считывает символ с клавиатуры и преобразует его в соответствующие направление, и void set_log(Log *log), который читает, какие уровни надо записывать через переданный указатель на объект наследника Log и записывает их в него.

UML-диаграмму межклассовых отношений см. в приложении А.

Выводы.

Реализован механизм логирования для игры на языке C++.

ПРИЛОЖЕНИЕ А

UML-ДИАГРАММА МЕЖКЛАССОВЫХ ОТНОШЕНИЙ