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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Полиморфизм

Студент гр. 3383 _____ Боривец С. Ю.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2024

Цель работы

Создать классы исключений, способностей, менеджера способностей, реализовать методы для взаимодействия с ними. Продумать архитектуру проекта, учитывая результаты предыдущей работы.

Задание

Лабораторная работа №2 - Полиморфизм

Создать класс-интерфейс способности, которую игрок может применять. Через наследование создать 3 разные способности:

Двойной урон - следующая атак при попадании по кораблю нанесет сразу 2 урона (уничтожит сегмент).

Сканер - позволяет проверить участок поля 2х2 клетки и узнать, есть ли там сегмент корабля. Клетки не меняют свой статус.

Обстрел - наносит 1 урон случайному сегменту случайного корабля. Клетки не меняют свой статус.

Создать класс менеджер-способностей. Который хранит очередь способностей, изначально игроку доступно по 1 способности в случайном порядке. Реализовать метод применения способности.

Реализовать функционал получения одной случайной способности при уничтожении вражеского корабля.

Реализуйте набор классов-исключений и их обработку для следующих ситуаций (можно добавить собственные):

Попытка применить способность, когда их нет

Размещение корабля вплотную или на пересечении с другим кораблем

Атака за границы поля

Примечания:

Интерфейс события должен быть унифицирован, чтобы их можно было единообразно использовать через интерфейс

Не должно быть явных проверок на тип данных

Выполнение работы

Первое, что требуется реализовать – класс-интерфейс способности, которую игрок может применять. Этот класс будет называться Ability, объявлен он в файле “ability.h”. Здесь прописывается два чистых виртуальных метода – use и print_activation.

Метод use использует способность, он принимает на вход координаты по которым будет использована способность, а также поле на котором будет использована способность.

Метод print_activation будет выводить сообщение, информирующее об активации способности.

Класс-интерфейс реализован, он унифицирован, в дальнейшем, для дальнейшего использования каждой способности через интерфейс, без явных проверок на тип данных.

Далее требуется создать три способности: двойной урон, сканер, обстрел.

Реализация двойного урона расположена в файлах “double_damage.h” и “double_damage.cpp”. Данная способность наследуется от класса интерфейса Ability. В случае активации способности, следующая атака нанесет 2 урона по сегменту корабля. Методы use и print_activation перегружаются, способность атакует два раза по требуемым координатам на переданном поле. Если координаты выходят за границу поля, то выбрасывается исключение.

Реализация сканера расположена в файлах “scanner.h” и “scanner.cpp”. Данная способность наследуется от класса интерфейса Ability. В случае активации способности, произойдет проверка поля 2x2 клетки, если там находятся сегменты корабля, то информация о них будет передана. Методы use и print_activation перегружаются, способность проверяет по требуемым координатам на переданном поле. Если координаты выходят за границу поля, соответствующая информация будет передана.

Реализация обстрела расположена в файлах “bombard.h” и “bombard.cpp”. Данная способность наследуется от класса интерфейса Ability.

В случае активации способности, случайному сегменту корабля нанесется урон. Методы use и print_activation перегружаются, способность собирает в вектор координаты всех поврежденных или нетронутых сегментов(с очками здоровья более нуля) корабля на поле, а затем случайным образом выбирает среди всех координат одну, и наносит в нее урон.

Класс менеджер-способностей. Реализован в файлах “ability_manager.h” и “ability_manager.cpp”. Имеет приватное поле – вектор указателей на Ability. Из методов присутствуют: конструктор, применение способности и получение случайной способности.

В конструкторе в вектор загружается по одной способности.

В методе применения способности apply_ability подаются координаты и игровое поле. Сначала запрашивается ответ пользователя, хочет ли он применить способность, а затем при положительном ответе, среди всех имеющихся способностей выбирается случайная и используется через интерфейс. Использованная способность удаляется.

В методе добавления случайной способности add_random_ability среди трех существующих способностей случайно выбирается одна и добавляется к имеющимся. Выводится сообщение о получении способности.

Для получения способности при уничтожении вражеского корабля, в корабле появилось поле destruction_flag, а также был реализован метод update_destruction_flag, который обновляет состояние корабля, возвращает true, если он стал разрушенным. В Ship_manager был реализован метод, который проверяет состояние всех кораблей и возвращает true, если хоть один корабль стал разрушенным. В таком случае с помощью add_random_ability в ability_manager будет добавлена случайная способность.

Набор классов-исключений находятся в папке exceptions, для каждого класса-исключения был создан свой .h и .cpp файл с таким же названием:

Attack_out_of_bound – атака за границы игрового поля.

Invalid_ability_usage – попытка использовать способности при их отсутствии.

Ship_placement_error – попытка поставить корабль вплотную или на пересечении с другим кораблем.

Все реализованные исключения были помещены в места кода, где ранее выводилось сообщение в терминал.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Создано поле, расставлены корабли на нем	Способности созданы и используются в случайном порядке. При уничтожении корабля приобретается новая способность	ОК

Выводы

Созданы классы исключений, способностей, менеджера способностей, реализованы методы для взаимодействия с ними. Архитектуру проекта продумана, результаты предыдущей работы учтены при ее создании.

ПРИЛОЖЕНИЕ А

UML-ДИАГРАММА КЛАССОВ