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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Объектно-ориентированное программирование»

Тема: Добавление класса управления игрой

Студент гр. 9381 _____ Колованов Р.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Изучить парадигму объектно-ориентрированного программирования; реализовать классы для управления игрой; изучить и реализовать паттерны проектирования *Command* и *Facade*.

Задание.

Создать класс игры, через который пользователь взаимодействует с игрой. Управление игроком, начало новой игры, завершение игры. Могут быть созданы дополнительные необходимые классы, которые отвечают отдельно за перемещение, создание игры и т.д. Но пользователь должен взаимодействовать через интерфейс одного класса.

Обязательные требования:

- Создан класс управления игрой;
- Взаимодействие сохраняет инвариант.

Дополнительные требования:

- Пользователь взаимодействует с использованием паттерна **Команды**;
- Взаимодействие с компонентами происходит через паттерн **Фасад**.

Выполнение работы.

Для начала были реализован класс управления игрой *GameController,* который предоставляет пользователю интерфейс для управления ходом игры. Класс *GameController* также представляет собой фасад для работы с подсистемой классов игрового поля и игровых объектов. Помимо этого, были реализованы интрефейс для команд *Command* и сами классы команд *StartLevelCommand*, *MovePlayerCommand* и *InteractPlayerCommand,* при помощи которых пользователь отправляет запросы объекту класса *GameController* на выполнение каких-либо действий.

В программе используются умные указатели, поэтому очистка памяти для них не требуется. Для реализации GUI-интерфейса программы был использован фреймворк *Qt*.

Подробное описание классов приведено ниже (см. Раздел *Описание классов и структур*).

Разработанный программный код см. в приложении А.

Описание классов и структур.

Класс GameController.

Представляет собой класс для управления игрой, помимо этого является фасадом, работающим подсистемой классов игрового поля и игровых объектов.

Поля класса *GameController*:

Модификатор доступа	Название и тип поля	Предназначение	Значение по умолчанию
private	*pPlayer player_*	Хранит адрес объекта игрока.	-
private	pLoggingListener loggingListener_	Хранит адрес объекта для отслеживания игровых объектов и вывод логов на консоль и в файл.	-
private	size_t level_	Хранит информацию о текущем уровне игры.	*0*
private	bool levelComplete_	Хранит информацию о том, завершил ли игрок текущий уровень.	*false*

Методы класса *GameController*:

Модификатор доступа	Возвращаемое значение	Название метода и принимаемые аргументы
public	-	*GameController(pLoggingListener& logger)*
public	*void*	*createLevel()*
public	*void*	*movePlayer(Direction direction)*
public	*void*	*executePlayerInteraction()*
public	*size_t*	*getLevelNumber()*
public	*pConstPlayer*	*getPlayer() const*
public	*void*	*getLevelPixmap(pQPixmap&*

		levelPixmap) const
public	*bool*	*isPlayerReachedExit() const*
public	*bool*	*isLevelComplete() const*
public	-	*~GameController()*

Класс Command.

Представляет собой интерфейс для классов команд.

Поля класса *Command*:

Модификатор доступа	Название и тип поля	Предназначение	Значение по умолчанию
protected	*pGameController controller_*	Хранит адрес объекта класса управления игрой, для которого предназначена команда.	-

Методы класса *Command*:

Модификатор доступа	Возвращаемое значение	Название метода и принимаемые аргументы
public	-	*Command(pGameController controller)*
public	*void*	*execute() = 0*
public	-	*~Command() = default*

Класс StartLevelCommand.

Представляет собой команду для создания уровня.

Методы класса *StartLevelCommand*:

Модификатор доступа	Возвращаемое значение	Название метода и принимаемые аргументы
public	-	*StartLevelCommand(pGameController*

		controller)
public	*void*	*execute()*

Класс MovePlayerCommand.

Представляет собой команду для перемещения персонажа.

Поля класса *MovePlayerCommand*:

Модификатор доступа	Название и тип поля	Предназначение	Значение по умолчанию
private	*Direction direction_*	Хранит неправление перемещения игрока.	-

Методы класса *MovePlayerCommand*:

Модификатор доступа	Возвращаемое значение	Название метода и принимаемые аргументы
public	-	*MovePlayerCommand(pGameController controller, Direction direction);*
public	*void*	*execute()*

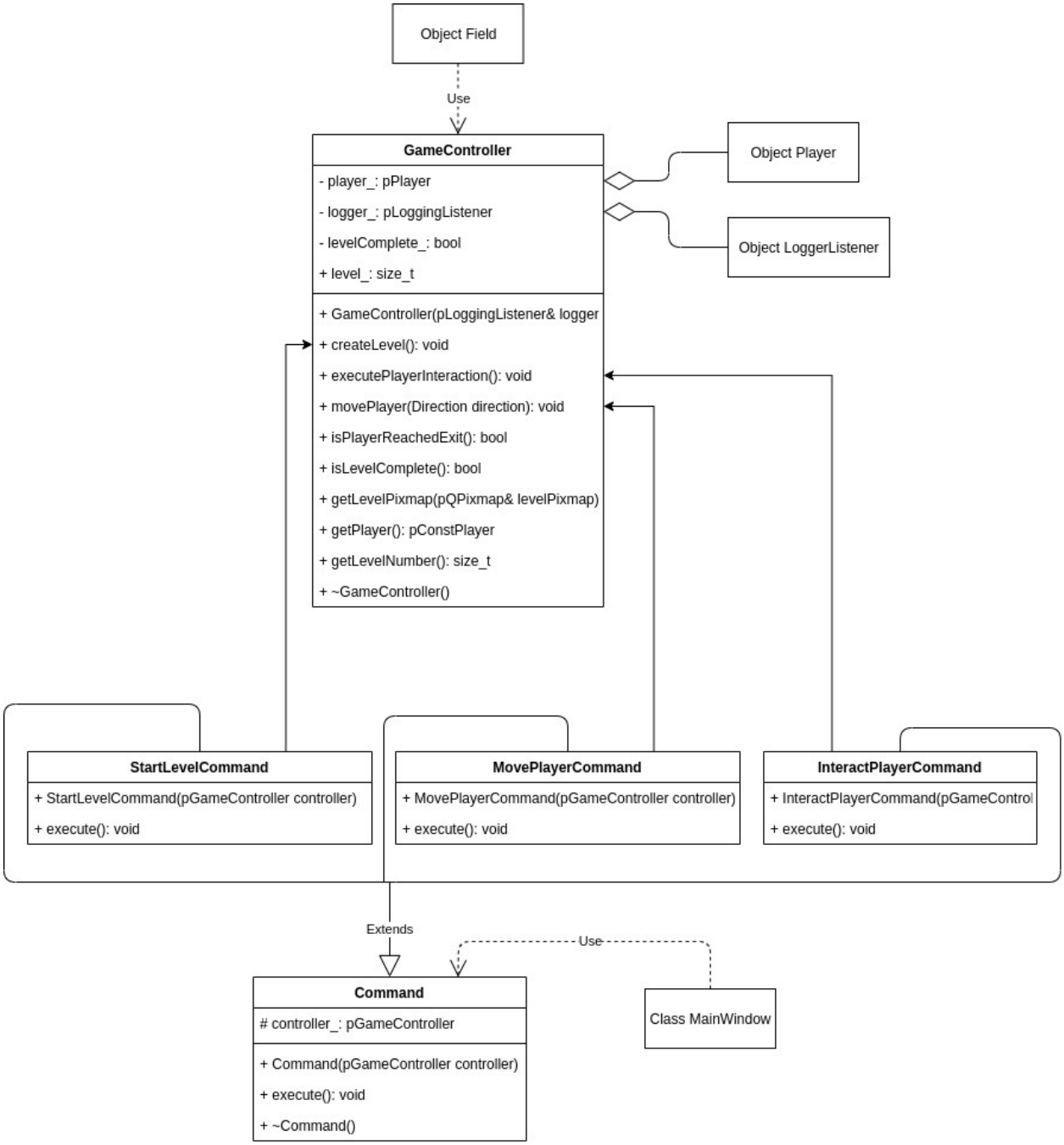
Класс InteractPlayerCommand.

Представляет собой команду для взаимодействия персонажа с объектом напротив него.

Методы класса *InteractPlayerCommand*:

Модификатор доступа	Возвращаемое значение	Название метода и принимаемые аргументы
public	-	*InteractPlayerCommand(pGameController controller);*
public	*void*	*execute()*

UML-диаграмма.



Тестирование.

Файл log.txt

```
     [10-11-20 09:25:59] Creating: Object of class 'Player':
Position((0, 0)); Health(73); MaxHealth(100); AttackDamage(3);
Protection(0); Rotation(3); PassFounded(0)
     [10-11-20 09:25:59] Creating the game field...
     [10-11-20 09:25:59] Object of class 'Player' change position to
[2, 1]
     [10-11-20 09:25:59] Creating the game field... Done.
     [10-11-20 09:26:00] Object of class 'Player' change position to
[2, 2]
     [10-11-20 09:26:00] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:00] Object of class 'Player' change position to
[3, 2]
     [10-11-20 09:26:00] Object of class 'Player' change rotation to
'Top'
     [10-11-20 09:26:00] Object of class 'Player' change position to
[3, 1]
     [10-11-20 09:26:01] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:01] Object of class 'Player' change rotation to
'Bottom'
     [10-11-20 09:26:01] Object of class 'Player' change position to
[3, 2]
     [10-11-20 09:26:01] Object of class 'Player' change position to
[3, 3]
     [10-11-20 09:26:02] Object of class 'Player' change position to
[3, 4]
     [10-11-20 09:26:02] Object of class 'Player' change position to
[3, 5]
     [10-11-20 09:26:02] Object of class 'Player' change position to
[3, 6]
     [10-11-20 09:26:03] Object of class 'Player' change rotation to
'Left'
     [10-11-20 09:26:03] Object of class 'Player' change position to
[2, 6]
     [10-11-20 09:26:03] Object of class 'Player' change position to
[1, 6]
     [10-11-20 09:26:03] Object of class 'Player' change rotation to
'Bottom'
     [10-11-20 09:26:03] Object of class 'Player' change position to
[1, 7]
     [10-11-20 09:26:03] Object of class 'Player' change position to
[1, 8]
     [10-11-20 09:26:03] Object of class 'Player' change position to
[1, 9]
     [10-11-20 09:26:04] Object of class 'Player' change position to
[1, 10]
     [10-11-20 09:26:04] Object of class 'Player' change position to
[1, 11]
     [10-11-20 09:26:04] Object of class 'Player' change position to
[1, 12]
```

```
     [10-11-20 09:26:05] Object of class 'Player' change position to
[1, 13]
     [10-11-20 09:26:05] Object of class 'Player' change position to
[1, 14]
     [10-11-20 09:26:05] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:05] Object of class 'Player' change position to
[2, 14]
     [10-11-20 09:26:05] Object of class 'Player' change rotation to
'Bottom'
     [10-11-20 09:26:05] Object of class 'Player' change position to
[2, 15]
     [10-11-20 09:26:05] Object of class 'Player' change position to
[2, 16]
     [10-11-20 09:26:06] Object of class 'Player' interact with
object of class '9Medicines'
     [10-11-20 09:26:06] Object of class 'Player' change health to 98
     [10-11-20 09:26:06] Destroying: Object of class 'Medicines':
HealthRecovery(25)
     [10-11-20 09:26:07] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:07] Object of class 'Player' change position to
[3, 16]
     [10-11-20 09:26:07] Object of class 'Player' change position to
[4, 16]
     [10-11-20 09:26:07] Object of class 'Player' change position to
[5, 16]
     [10-11-20 09:26:07] Object of class 'Player' change position to
[6, 16]
     [10-11-20 09:26:08] Object of class 'Player' change position to
[7, 16]
     [10-11-20 09:26:08] Object of class 'Player' change position to
[8, 16]
     [10-11-20 09:26:08] Object of class 'Player' change position to
[9, 16]
     [10-11-20 09:26:08] Object of class 'Player' change position to
[10, 16]
     [10-11-20 09:26:08] Object of class 'Player' change rotation to
'Top'
     [10-11-20 09:26:09] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:09] Object of class 'Player' change position to
[11, 16]
     [10-11-20 09:26:09] Object of class 'Player' change position to
[12, 16]
     [10-11-20 09:26:09] Object of class 'Player' change rotation to
'Top'
     [10-11-20 09:26:09] Object of class 'Player' change position to
[12, 15]
     [10-11-20 09:26:09] Object of class 'Player' change position to
[12, 14]
     [10-11-20 09:26:10] Object of class 'Player' change position to
[12, 13]
     [10-11-20 09:26:10] Object of class 'Player' change position to
[12, 12]
```

```
     [10-11-20 09:26:10] Object of class 'Player' change position to
[12, 11]
     [10-11-20 09:26:10] Object of class 'Player' change position to
[12, 10]
     [10-11-20 09:26:11] Object of class 'Player' change position to
[12, 9]
     [10-11-20 09:26:11] Object of class 'Player' change position to
[12, 8]
     [10-11-20 09:26:11] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:11] Object of class 'Player' change position to
[13, 8]
     [10-11-20 09:26:11] Object of class 'Player' change position to
[14, 8]
     [10-11-20 09:26:11] Object of class 'Player' change position to
[15, 8]
     [10-11-20 09:26:12] Object of class 'Player' change position to
[16, 8]
     [10-11-20 09:26:12] Object of class 'Player' change position to
[17, 8]
     [10-11-20 09:26:12] Object of class 'Player' interact with
object of class '15LevelPassObject'
     [10-11-20 09:26:12] Destroying: Object of class
'LevelPassObject'.
     [10-11-20 09:26:12] Object of class 'Player' change rotation to
'Left'
     [10-11-20 09:26:12] Object of class 'Player' change position to
[16, 8]
     [10-11-20 09:26:12] Object of class 'Player' change position to
[15, 8]
     [10-11-20 09:26:13] Object of class 'Player' change position to
[14, 8]
     [10-11-20 09:26:13] Object of class 'Player' change rotation to
'Top'
     [10-11-20 09:26:13] Object of class 'Player' change rotation to
'Left'
     [10-11-20 09:26:13] Object of class 'Player' change position to
[13, 8]
     [10-11-20 09:26:14] Object of class 'Player' change rotation to
'Top'
     [10-11-20 09:26:14] Object of class 'Player' change position to
[13, 7]
     [10-11-20 09:26:14] Object of class 'Player' change position to
[13, 6]
     [10-11-20 09:26:14] Object of class 'Player' change position to
[13, 5]
     [10-11-20 09:26:14] Object of class 'Player' change position to
[13, 4]
     [10-11-20 09:26:14] Object of class 'Player' change position to
[13, 3]
     [10-11-20 09:26:15] Object of class 'Player' change position to
[13, 2]
     [10-11-20 09:26:15] Object of class 'Player' change rotation to
'Right'
     [10-11-20 09:26:15] Object of class 'Player' change position to
```

```
[14, 2]
     [10-11-20 09:26:15] Object of class 'Player' change position to
[15, 2]
     [10-11-20 09:26:15] Object of class 'Player' change position to
[16, 2]
     [10-11-20 09:26:15] Object of class 'Player' change position to
[17, 2]
     [10-11-20 09:26:16] Object of class 'Player' change position to
[18, 2]
     [10-11-20 09:26:16] Game over! Player has reached the end of the
level.
     [10-11-20 09:26:16] Destroying: Object of class 'Weapon':
Damage(8)
     [10-11-20 09:26:16] Destroying: Object of class 'Armor':
ProtectionValue(7)
     [10-11-20 09:26:16] Destroying: Object of class 'Weapon':
Damage(10)
     [10-11-20 09:26:16] Destroying: Object of class 'Armor':
ProtectionValue(10)
     [10-11-20 09:26:16] Destroying: Object of class 'Medicines':
HealthRecovery(25)
     [10-11-20 09:26:16] Quitting the game…
```

Выводы.

Была изучена парадигма объектно-ориентрированного программирования. Были реализованы класс управления игрой, интерфейс для классов команд *Command* и сами классы команд *StartLevelCommand*, *MovePlayerCommand* и *InteractPlayerCommand*. Были изучены и реализованы паттерны проектирования *Command* и *Facade*. По мимо этого, был реализован GUI-интерфейс игры при помощи фреймворка Qt.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: GameController.h

```
#ifndef GAMECONTROLLER_H
#define GAMECONTROLLER_H

#include <QPixmap>
#include "classes/game/objects/creatures/player/player.h"
#include "classes/game/field.h"
#include "classes/logging/logginglistener.h"

typedef std::shared_ptr<class GameController> pGameController;
typedef std::shared_ptr<QPixmap> pQPixmap;


class GameController {
private:
    pPlayer player_;
    pLoggingListener logger_;
    size_t level_ = 0;
    bool levelComplete_ = false;

public:
    GameController(pLoggingListener& logger);
    void createLevel();
    void movePlayer(Direction direction);
    void executePlayerInteraction();
    bool isPlayerReachedExit() const;
    bool isLevelComplete() const;
    void getLevelPixmap(pQPixmap& levelPixmap) const;
    pConstPlayer getPlayer();
    size_t getLevelNumber();
    ~GameController();
};


#endif // GAMECONTROLLER_H
```

Название файла: GameController.cpp

```
#include <QMap>
#include "classes/game/gamecontroller.h"
#include "classes/game/levelgenerator.h"
#include "classes/gui/levelpainter.h"

GameController::GameController(pLoggingListener& logger):
logger_(logger) {
    pEventListener listener(logger);

    player_ = pPlayer(new Player(Position2D(0, 0)));
    player_->setMaxHealth(100);
    player_->setHealth(73);
    player_->setAttackDamage(3);
    player_->setProtection(0);
    player_->getEventManager().subscribe(listener);
}
```

```
      void GameController::createLevel() {
          LevelGenerator levelGenerator(logger_);
          levelGenerator.generate(Size2D(2 + level_, 2 + level_), level_);
          levelComplete_ = false;
          player_->setPosition(levelGenerator.getEntryPosition());
          player_->setPassFounded(false);
      }

      bool GameController::isPlayerReachedExit() const {
          const Field& field = Field::getInstance();
          return field.getCell(player_->getPosition()).getType() ==
CellType::Exit;
      }

      void GameController::movePlayer(Direction direction) {
          Field& field = Field::getInstance();
          Position2D newPosition = Position2D(player_->getPosition().x,
player_->getPosition().y);

          newPosition.shift(direction);
          player_->setRotation(direction);

          if (field.getCell(newPosition).isPassable()) {
              player_->setPosition(newPosition);
          }

          if (isPlayerReachedExit() && player_->getPassFounded()) {
              levelComplete_ = true;
              level_++;
          }
      }

      void GameController::executePlayerInteraction() {
          Field& field = Field::getInstance();
          Position2D interactionPosition = player_-
>getPosition().shift(player_->getRotation());;
          *player_ <= field.getCell(interactionPosition).getObject();
      }

      bool GameController::isLevelComplete() const {
          return levelComplete_;
      }

      void GameController::getLevelPixmap(pQPixmap& levelPixmap) const {
          LevelPainter::paint(levelPixmap, player_);
      }

      pConstPlayer GameController::getPlayer() {
          return player_;
      }

      size_t GameController::getLevelNumber() {
          return level_;
      }

      GameController::~GameController() {
          logger_->update("Quitting the game...\n");
      }
```

Название файла: Command.h

```
#ifndef COMMAND_H
#define COMMAND_H

#include "classes/game/gamecontroller.h"

typedef std::shared_ptr<class Command> pCommand;


class Command {
protected:
    pGameController controller_;

public:
    Command(pGameController controller);
    virtual void execute() = 0;
    virtual ~Command() = default;
};


#endif // COMMAND_H
```

Название файла: InteractPlayerCommand.h

```
#ifndef INTERACT_PLAYER_COMMAND_H
#define INTERACT_PLAYER_COMMAND_H

#include "classes/game/commands/command.h"


class InteractPlayerCommand: public Command {
public:
    InteractPlayerCommand(pGameController controller);
    void execute();
};


#endif // INTERACT_PLAYER_COMMAND_H
```

Название файла: MovePlayerCommand.h

```
#ifndef MOVE_PLAYER_COMMAND_H
#define MOVE_PLAYER_COMMAND_H

#include "classes/game/commands/command.h"
#include "classes/enumerations/direction.h"


class MovePlayerCommand: public Command {
private:
    Direction direction_;

public:
    MovePlayerCommand(pGameController controller, Direction direction);
    void execute();
};

#endif // MOVE_PLAYER_COMMAND_H
```

Название файла: StartLevelCommand.h

```
#ifndef START_LEVEL_COMMAND_H
#define START_LEVEL_COMMAND_H

#include "classes/game/commands/command.h"


class StartLevelCommand: public Command {
public:
    StartLevelCommand(pGameController controller);
    void execute();
};


#endif // START_LEVEL_COMMAND_H
```

Название файла: Command.cpp

```
#include "command.h"

Command::Command(pGameController controller): controller_(controller) {}
```

Название файла: InteractPlayerCommand.cpp

```
#include "interactplayercommand.h"

InteractPlayerCommand::InteractPlayerCommand(pGameController
controller): Command(controller) {}

void InteractPlayerCommand::execute() {
    controller_->executePlayerInteraction();
}
```

Название файла: MovePlayerCommand.cpp

```
#include "moveplayercommand.h"

MovePlayerCommand::MovePlayerCommand(pGameController controller,
Direction direction): Command(controller), direction_(direction) {}

void MovePlayerCommand::execute() {
    controller_->movePlayer(direction_);
}
```

Название файла: StartLevelCommand.cpp

```
#include "startlevelcommand.h"

StartLevelCommand::StartLevelCommand(pGameController controller):
Command(controller) {}

void StartLevelCommand::execute() {
    controller_->createLevel();
}
```

Название файла: Command.h

G

Название файла: Command.h

G

Название файла: Command.h

G