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

ОТЧЁТ

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

Тема: Сериализация состояния программы

Студент гр. 8303 _____ Парфентьев Л.М.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы

Добавить в программу сериализацию и восстановление состояния.

Задание

Реализация сохранения и загрузки состояния программы. Основные требования:

- Возможность записать состояние программы в файл
- Возможность считать состояние программы из файла

Ход выполнения работы

- Объекты сохраняются следующим образом: выводится строковое обозначение класса объекта, а затем его параметры.
- При восстановлении объектов считывается строка, и по ней ищется объект, отвечающий за создание основы объекта восстанавливаемого объекта (`Restorer`). Объект `Restorer` может считывать дополнительную информацию прежде чем создавать объект (например, размер поля). Затем созданный объект сам считывает свои параметры.
- В сериализации задействованы следующие классы:
  - `Storable`: сериализуемый объект;
  - `Restorer`: создаёт восстанавливаемый объект;
  - `RestorerTable`: таблица объектов `Restorer`. Ключи – обозначения классов (`std::string`).
- Некоторые объекты могут содержать переменное количество других объектов. Для считывания таких объектов создан класс `StorableEnd`. При его сериализации выводится слово „end“. Считывание такого объекта исользуется как маркер конца ввода содержимого другого объекта.

- Для объектов, расположенных в другом объекте, созданы классы сериализации/десериализации объектов с индексами или координатами.
- Для сериализации координат создан класс `StorableCoordinates`.

Выводы

Реализована сериализация и десериализация состояния программы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: storable.hpp

```
#ifndef _STORABLE_HPP
#define _STORABLE_HPP

#include <iostream>
#include <map>
#include <string>
#include <utility>


class RestorerTable;

class Storable {
public:
    virtual void store(std::ostream &os) const =0;
    virtual bool restore(std::istream &,
                         RestorerTable *) { return true; };

    virtual ~Storable() {}
};

#define TRIVIALLY_STORABLE(keyword)             \
    public:                                     \
    virtual void                                \
    store(std::ostream &os) const override      \
    {                                           \
        os << keyword "\n";                     \
    }




class Restorer {
public:
    virtual Storable *restore(std::istream &is,
                              RestorerTable *tab) const =0;
};

class RestorerTable {
    std::map<std::string, Restorer *> _tab;

public:
    RestorerTable(std::map<std::string, Restorer *> m)
        :_tab{std::move(m)} {}

    Storable *
```

```
    restore(std::istream &is)
    {
        std::string n;
        is >> n;

        auto iter = _tab.find(n);
        if (iter == _tab.end()) {
            return nullptr;
        }

        Storable *s = iter->second->restore(is, this);
        if (!s) {
            return nullptr;
        }

        if (!s->restore(is, this)) {
            delete s;
            return nullptr;
        }
        return s;
    }

    static RestorerTable *defaultTable();
};

#endif
```

Название файла: storable.cpp

```
#include <iostream>
#include <map>
#include <string>

#include "storable.hpp"
#include "restorers.hpp"
#include "common_storables.hpp"

#include "melee_units.hpp"
#include "ranged_units.hpp"
#include "catapult_units.hpp"
#include "landscape_types.hpp"
#include "neutral_object_types.hpp"

#include "factory_table.hpp"
#include "iostream_player.hpp"


RestorerTable *
RestorerTable::defaultTable()
{
    return new RestorerTable {{
```

```
{"end", new SimpleRestorer<StorableEnd> {}},
{"coords", new SimpleRestorer<StorableCoordinates> {}},
{"index", new SimpleRestorer<StorableWithIndex> {}},
{"at", new SimpleRestorer<StorableWithCoords> {}},

{"game", new restorers::GameRestorer {}},
{"map", new restorers::MapRestorer {}},

{"base", new SimpleRestorer<Base> {}},

{"iostream_player", new SimpleRestorer<IostreamPlayer> {}},

{"l_normal", new SimpleRestorer<landscapes::Normal> {}},
{"l_swamp", new SimpleRestorer<landscapes::Swamp> {}},
{"l_forest", new SimpleRestorer<landscapes::Forest> {}},

{"u_swordsman", new SimpleRestorer<units::Swordsman> {}},
{"u_spearsman", new SimpleRestorer<units::Spearsman> {}},
{"u_cavalry", new SimpleRestorer<units::Cavalry> {}},

{"u_archer", new SimpleRestorer<units::Archer> {}},
{"u_slinger", new SimpleRestorer<units::Slinger> {}},

{"u_onager", new SimpleRestorer<units::Onager> {}},
{"u_boltthrower", new SimpleRestorer<units::BoltThrower> {}},

{"n_healingwell", new SimpleRestorer<objects::HealingWell> {}},
{"n_tower", new SimpleRestorer<objects::Tower> {}},
{"n_tunnelentrance", new SimpleRestorer<objects::TunnelsEntrance> {}},
{"n_weaponsmiths", new SimpleRestorer<objects::WeaponSmiths> {}},

{"uf_melee",
 new SimpleRestorer<objects::WeaponSmiths::MeleeUnitFilter> {}},
{"uf_ranged",
 new SimpleRestorer<objects::WeaponSmiths::RangedUnitFilter> {}},
{"uf_catapult",
 new SimpleRestorer<objects::WeaponSmiths::CatapultUnitFilter> {}},

{"mp_basic", new SimpleRestorer<BasicMovement> {}},
{"mp_modifyiing", new SimpleRestorer<ModifyingMovePolicy> {}},

{"dp_basic", new SimpleRestorer<BasicDefense> {}},
{"dp_level_deco", new SimpleRestorer<DefenseLevelDeco> {}},
{"dp_multiplier", new SimpleRestorer<MultiplierDefensePolicy> {}},

{"ap_forbidden", new SimpleRestorer<AttackForbidden> {}},
{"ap_multiplier", new SimpleRestorer<MultiplierAttackPolicy> {}},
{"ap_extended", new SimpleRestorer<ExtendedShootingRange> {}},

{"ap_melee", new SimpleRestorer<MeleeAttack> {}},
{"ap_ranged", new SimpleRestorer<RangedAttack> {}},
{"ap_catapult", new SimpleRestorer<CatapultAttack> {}},
```

```
    }};
}
```

Название файла: common_storables.hpp

```
#ifndef _H_COMMON_STORABLES_HPP
#define _H_COMMON_STORABLES_HPP

#include "storable.hpp"
#include "object_print.hpp"


class StorableEnd: public Storable {
    TRIVIALLY_STORABLE("end");
};

class StorableCoordinates: public Storable {
    Vec2 _c;

public:
    StorableCoordinates() {}

    StorableCoordinates(Vec2 c) :_c{c} {}

    virtual void
    store(std::ostream &os) const override
    {
        os << "coords " << _c.x() << " " << _c.y() << "\n";
    }

    virtual bool
    restore(std::istream &is,
            RestorerTable *) override
    {
        int x, y;
        is >> x >> y;
        _c = Vec2{x, y};
        return !is.fail();
    }

    Vec2 coords() const { return _c; }
};

class StorableWithIndex: public Storable {
    int _i;
    Storable *_s;

public:
    StorableWithIndex() {}

    StorableWithIndex(int idx, Storable *s)
```

```
        :_i{idx}, _s{s} {}

    virtual void
    store(std::ostream &os) const override
    {
        os << "index " << _i << " ";
        _s->store(os);
    }

    virtual bool
    restore(std::istream &is,
            RestorerTable *tab) override
    {
        is >> _i;
        _s = tab->restore(is);
        return !is.fail() && _s;
    }

    int index() const { return _i; }
    Storable *child() const { return _s; }

    static void
    storeWithIndex(int idx, const Storable *s,
                   std::ostream &os)
    {
        os << "index " << idx << " ";
        s->store(os);
    }
};

class StorableWithCoords: public Storable {
    Vec2 _c;
    Storable *_s;

public:
    StorableWithCoords() {}

    StorableWithCoords(Vec2 c, Storable *s)
        :_c{c}, _s{s} {}

    virtual void
    store(std::ostream &os) const override
    {
        os << "at " << _c.x() << " " << _c.y() << " ";
        _s->store(os);
    }

    virtual bool
    restore(std::istream &is,
            RestorerTable *tab) override
    {
        int x, y;
```

```
        is >> x >> y;
        _c = Vec2{x, y};
        _s = tab->restore(is);
        return !is.fail() && _s;
    }

    Vec2 coords() const { return _c; }
    Storable *child() const { return _s; }

    static void
    storeWithCoords(Vec2 pt, const Storable *s,
                    std::ostream &os)
    {
        os << "at " << pt.x() << " " << pt.y() << " ";
        s->store(os);
    }
};

#endif
```

Название файла: restorers.hpp

```
#ifndef _H_RESTORERS_HPP
#define _H_RESTORERS_HPP

#include "storable.hpp"

#include "point.hpp"
#include "map.hpp"
#include "unit.hpp"

#include "base.hpp"
#include "game.hpp"
#include "player.hpp"


template<typename T>
class SimpleRestorer: public Restorer {
public:
    virtual Storable *
    restore(std::istream &,
            RestorerTable *) const override
    {
        return new T {};
    }
};


namespace restorers {

    class GameRestorer: public Restorer {
    public:
```

```
        virtual Storable *
        restore(std::istream &is,
                RestorerTable *tab) const override
        {
            Storable *s = tab->restore(is);
            Map *map = dynamic_cast<Map *>(s);
            if (!map) {
                delete s;
                return nullptr;
            }

            return new Game {map};
        }
    };

    class MapRestorer: public Restorer {
    public:
        virtual Storable *
        restore(std::istream &is,
                RestorerTable *) const override
        {
            int w, h;
            is >> w >> h;

            return new Map {w, h};
        }
    };

}

#endif
```