Back End

StatTools.h

#pragma once

#include "DailyReport.h"

#include <string>

namespace StatTools {

template <class T>

double statistic(DailyReport\* target) {

double total = 0;

size\_t number = 0;

for (size\_t i = 0; i < target->getReturnsSize(); i++) {

if (typeid(T) == typeid(int)) {

total += target->getReturn(i)->getDisk().getLength();

number++;

}

if (typeid(T) == typeid(std::wstring)) {

total += target->getReturn(i)->getDisk().getName().length();

number++;

}

}

for (size\_t i = 0; i < target->getExtraditionsSize(); i++) {

if (typeid(T) == typeid(int)) {

total += target->getExtradition(i)->getDisk().getLength();

number++;

}

if (typeid(T) == typeid(std::wstring)) {

total += target->getExtradition(i)->getDisk().getName().length();

number++;

}

}

if (!number) {

return 0;

}

return total / number;

}

}

Abonent.h

#pragma once

#include "Date1.h"

#include "Nameable.h"

#include "FWString.h"

class Abonent : public Date1, public Nameable{

protected:

FWString surename;

public:

Abonent(const FWString firstname, const FWString surename, const short year, const short month) : Date1(year, month), Nameable(firstname), surename(surename) {}

Abonent() : Date1(), Nameable(), surename(L"") {}

Abonent(const Abonent& reference);

~Abonent() {}

void setSurename(const FWString surename);

FWString getSurename() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator=(const Abonent& reference);

};

Abonent.cpp

#include "Abonent.h"

Abonent::Abonent(const Abonent& reference) : Date1(reference), Nameable(reference) {

this->surename = reference.surename;

}

void Abonent::setSurename(const FWString surename) {

this->surename = surename;

}

FWString Abonent::getSurename() const {

return this->surename;

}

void Abonent::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Abonent).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

this->name.save(stream);

this->surename.save(stream);

Date1::save(stream);

}

void Abonent::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Abonent).hash\_code()) {

throw WrongInputFileException();

}

this->name.load(stream);

this->surename.load(stream);

Date1::load(stream);

}

MagicJSON::JsonObject Abonent::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"abonent");

json.addInteger(L"\_\_hash", typeid(Abonent).hash\_code());

json.addString(L"name", this->name);

json.addString(L"surename", this->surename);

json.addObject(L"date", Date1::serialize());

return json;

}

void Abonent::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"abonent") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

this->surename = json.getString(L"surename");

Date1::deserialize(json.getObject(L"date"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Abonent::operator=(const Abonent& reference) {

this->surename = reference.surename;

Date1::operator=(reference);

Nameable::operator=(reference);

}

DailyReport.h

#pragma once

#include "Object.h"

#include "FVector.h"

#include "Operation.h"

#include "DailyReportException.h"

class DailyReport : public Object, public Fileable, public Serializeable{

protected:

FVector returns;

FVector extraditions;

public:

DailyReport();

~DailyReport() {}

DailyReport(const DailyReport& reference);

void addReturn(Operation\* return\_);

Operation\* getReturn(const size\_t index) const;

size\_t getReturnsSize();

void addExtradition(Operation\* extradition);

Operation\* getExtradition(const size\_t index) const;

size\_t getExtraditionsSize();

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

Operation find(const wstring name) const;

Operation find(const short year) const;

public:

void operator= (const DailyReport& reference);

};

DailyReport.cpp

#include "DailyReport.h"

DailyReport::DailyReport() {

this->extraditions.addObjectCreator((hash\_code)typeid(Operation).hash\_code(), []() -> Fileable \* { return new Operation(); });

this->returns.addObjectCreator((hash\_code)typeid(Operation).hash\_code(), []() -> Fileable \* { return new Operation(); });

}

DailyReport::DailyReport(const DailyReport& reference) : Object(reference) {

this->returns = reference.returns;

this->extraditions = reference.extraditions;

}

void DailyReport::addReturn(Operation\* return\_) {

if (!return\_->isReturn()) {

throw InputIsNotReturn();

}

this->returns.push\_back(return\_);

}

Operation\* DailyReport::getReturn(const size\_t index) const {

Operation\* temp = dynamic\_cast<Operation\*>(this->returns.get(index));

if (!temp) {

throw ZeroPointerException();

}

return temp;

}

size\_t DailyReport::getReturnsSize() {

return this->returns.size();

}

void DailyReport::addExtradition(Operation\* extradition) {

if (extradition->isReturn()) {

throw InputIsNotExtradition();

}

this->extraditions.push\_back(extradition);

}

Operation\* DailyReport::getExtradition(const size\_t index) const {

Operation\* temp = dynamic\_cast<Operation\*>(this->extraditions.get(index));

if (!temp) {

throw ZeroPointerException();

}

return temp;

}

size\_t DailyReport::getExtraditionsSize() {

return this->extraditions.size();

}

void DailyReport::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(DailyReport).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

this->extraditions.save(stream);

this->returns.save(stream);

}

void DailyReport::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(DailyReport).hash\_code()) {

throw WrongInputFileException();

}

this->extraditions.load(stream);

this->returns.load(stream);

}

MagicJSON::JsonObject DailyReport::serialize() {

MagicJSON::JsonObject json;

MagicJSON::JsonArray jreturns;

MagicJSON::JsonArray jextraditions;

for (Fileable\* item : this->returns) {

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) != nullptr) {

jreturns.addObject(serializeable->serialize());

}

}

for (Fileable\* item : this->extraditions) {

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) != nullptr) {

jextraditions.addObject(serializeable->serialize());

}

}

json.addString(L"\_\_type", L"dailyreport");

json.addInteger(L"\_\_hash", typeid(DailyReport).hash\_code());

json.addArray(L"returns", jreturns);

json.addArray(L"extraditions", jextraditions);

return json;

}

void DailyReport::deserialize(MagicJSON::JsonObject json) {

try {

this->returns.clear();

this->extraditions.clear();

wstring type = json.getString(L"\_\_type");

if (type.compare(wstring(L"dailyreport"))) {

throw UnexpectedTypeException();

}

MagicJSON::JsonArray jreturns = json.getArray(L"returns");

MagicJSON::JsonArray jextraditions = json.getArray(L"extraditions");

for (size\_t i = 0; i < jreturns.size(); i++) {

MagicJSON::JsonObject object = jreturns.getObject(i);

long hash = object.getInteger(L"\_\_hash");

size\_t s\_hash = (size\_t)hash;

Fileable\* item = this->returns.getObjectCreator(s\_hash)();

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) == nullptr) {

throw IncorrectObjectDataException();

}

serializeable->deserialize(object);

this->returns.push\_back(item);

}

for (size\_t i = 0; i < jextraditions.size(); i++) {

MagicJSON::JsonObject object = jextraditions.getObject(i);

Fileable\* item = this->returns.getObjectCreator(object.getInteger(L"\_\_hash"))();

Serializeable\* serializeable = nullptr;

if ((serializeable = dynamic\_cast<Serializeable\*>(item)) == nullptr) {

throw IncorrectObjectDataException();

}

serializeable->deserialize(object);

this->extraditions.push\_back(item);

}

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

Operation DailyReport::find(const wstring name) const {

for (Fileable\* item : this->returns) {

Operation\* operation = dynamic\_cast<Operation\*>(item);

if (operation) {

if (operation->getAbonent().getName() == name) {

return \*operation;

}

}

}

throw NoObjectFoundException();

}

Operation DailyReport::find(const short year) const {

for (Fileable\* item : this->returns) {

Operation\* operation = dynamic\_cast<Operation\*>(item);

if (operation) {

if (operation->getAbonent().getYear() == year) {

return \*operation;

}

}

}

throw NoObjectFoundException();

}

void DailyReport::operator=(const DailyReport& reference) {

this->returns = reference.returns;

this->extraditions = reference.extraditions;

}

Date1.h

#pragma once

#include "Object.h"

#include "Fileable.h"

#include "Exception.h"

#include "Serializeable.h"

class Date1 : public Object, public Fileable, public Serializeable {

protected:

short year;

short month;

public:

Date1(const short year, const short month);

Date1() : Date1(2000, 1) {}

~Date1() {}

Date1(const Date1& reference);

short getYear() const;

void setYear(const short year);

short getMonth() const;

void setMonth(const short year);

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Date1& reference);

bool operator> (const Date1& right);

bool operator< (const Date1& right);

bool operator>= (const Date1& right);

bool operator<= (const Date1& right);

bool operator== (const Date1& right);

};

Date1.cpp

#include "Date1.h"

Date1::Date1(const short year, const short month) : Object() {

if (year < 0 || month <= 0 || month > 12) {

throw WrongInputValuesException();

}

this->year = year;

this->month = month;

}

Date1::Date1(const Date1& reference) : Object(reference) {

this->year = reference.year;

this->month = reference.month;

}

short Date1::getYear() const {

return this->year;

}

void Date1::setYear(const short year) {

if (year < 0) {

throw WrongInputValuesException();

}

this->year = year;

}

short Date1::getMonth() const {

return this->month;

}

void Date1::setMonth(const short month) {

if (month < 0 || month > 11) {

throw WrongInputValuesException();

}

this->month = month;

}

void Date1::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Date1).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*)&(this->year), sizeof(short));

stream.write((char\*)&(this->month), sizeof(short));

}

void Date1::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*) & hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Date1).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->year), sizeof(short));

stream.read((char\*) & (this->month), sizeof(short));

}

MagicJSON::JsonObject Date1::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"date");

json.addInteger(L"\_\_hash", typeid(Date1).hash\_code());

json.addInteger(L"month", (long)this->month);

json.addInteger(L"year", (long)this->year);

return json;

}

void Date1::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"date") {

throw UnexpectedTypeException();

}

this->month = (short)json.getInteger(L"month");

this->year = (short)json.getInteger(L"year");

}

catch (MagicJSON::NoObjectFoundException e){

throw IncorrectObjectDataException();

}

}

void Date1::operator=(const Date1& reference) {

this->year = reference.year;

this->month = reference.month;

}

bool Date1::operator>(const Date1& right) {

if (this->year > right.year && this->month > right.month) {

return true;

}

return false;

}

bool Date1::operator<(const Date1& right) {

if (this->year < right.year && this->month < right.month) {

return true;

}

return false;

}

bool Date1::operator>=(const Date1& right) {

if (this->year >= right.year && this->month >= right.month) {

return true;

}

return false;

}

bool Date1::operator<=(const Date1& right) {

if (this->year <= right.year && this->month <= right.month) {

return true;

}

return false;

}

bool Date1::operator==(const Date1& right) {

if (this->year == right.year && this->month == right.month) {

return true;

}

return false;

}

Date2.h

#pragma once

#include "Date1.h"

class Date2 : public Date1 {

protected:

short day;

public:

Date2(const short year, const short month, const short day); // TODO: Finish exceptions when 31 and 30 days

Date2() : Date1(), day(0) {}

~Date2() {}

Date2(const Date2& reference);

short getDay() const;

void setDay(const short day);

public:

void save(ofstream & stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator=(const Date2& reference);

bool operator> (const Date2& right);

bool operator< (const Date2& right);

bool operator>= (const Date2& right);

bool operator<= (const Date2& right);

bool operator== (const Date2& right);

};

Date2.cpp

#include "Date2.h"

Date2::Date2(const short year, const short month, const short day) : Date1(year, month) {

this->setDay(day);

}

Date2::Date2(const Date2& reference) : Date1(reference) {

this->day = reference.day;

}

short Date2::getDay() const {

return this->day;

}

void Date2::setDay(const short day) {

if (day < 0 || day > 31) {

throw WrongInputValuesException();

}

if (this->month % 2 && day > 30) {

throw WrongInputValuesException();

}

if (this->month == 11 && (day > 28)) {

throw WrongInputValuesException();

}

this->day = day;

}

void Date2::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Date2).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*) & (this->day), sizeof(short));

Date1::save(stream);

}

void Date2::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)&hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Date2).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->day), sizeof(short));

Date1::load(stream);

}

void Date2::operator=(const Date2& reference) {

this->day = reference.day;

Date1::operator=(reference);

}

bool Date2::operator>(const Date2& right) {

if (this->day > right.day && this->operator>(right)) {

return true;

}

return false;

}

bool Date2::operator<(const Date2& right) {

if (this->day < right.day && this->operator<(right)) {

return true;

}

return false;

}

bool Date2::operator>=(const Date2& right) {

if (this->day >= right.day && this->operator>=(right)) {

return true;

}

return false;

}

bool Date2::operator<=(const Date2& right) {

if (this->day <= right.day && this->operator<=(right)) {

return true;

}

return false;

}

bool Date2::operator==(const Date2& right) {

if (this->day == right.day && this->operator==(right)) {

return true;

}

return false;

}

MagicJSON::JsonObject Date2::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"date");

json.addInteger(L"\_\_hash", typeid(Date2).hash\_code());

json.addInteger(L"month", (long)this->month);

json.addInteger(L"year", (long)this->year);

json.addInteger(L"day", (long)this->day);

return json;

}

void Date2::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"date") {

throw UnexpectedTypeException();

}

this->month = (short)json.getInteger(L"month");

this->year = (short)json.getInteger(L"year");

this->day = (short)json.getInteger(L"day");

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

Disk.h

#pragma once

#include "Date2.h"

#include "String.h"

#include "Studio.h"

#include "Nameable.h"

class Disk : public Date2, public Nameable {

protected:

unsigned int length;

Studio studio;

public:

Disk(const Studio studio, const unsigned int length, const FWString name, const short year, const short month, const short day) :

Date2(year, month, day), Nameable(name), studio(studio), length(length) {}

Disk() : Date2(), Nameable(), length(0), studio(Studio()) {}

Disk(const Disk& reference);

~Disk() {}

void setLength(const unsigned int length);

unsigned int getLength() const;

void setStudio(const Studio studio);

Studio getStudio() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Disk& reference);

};

Disk.cpp

#include "Disk.h"

Disk::Disk(const Disk& reference) : Date2(reference), Nameable(reference) {

this->studio = reference.studio;

this->length = reference.length;

}

void Disk::setLength(const unsigned int length) {

this->length = length;

}

unsigned int Disk::getLength() const {

return this->length;

}

void Disk::setStudio(const Studio studio) {

this->studio = studio;

}

Studio Disk::getStudio() const {

return this->studio;

}

void Disk::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Disk).hash\_code();

stream.write((char\*)&hash, sizeof(hash\_code));

stream.write((char\*) & (this->length), sizeof(unsigned int));

this->studio.save(stream);

Date2::save(stream);

this->name.save(stream);

}

void Disk::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)&hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Disk).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*) & (this->length), sizeof(unsigned int));

this->studio.load(stream);

Date2::load(stream);

this->name.load(stream);

}

MagicJSON::JsonObject Disk::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"disk");

json.addInteger(L"\_\_hash", typeid(Disk).hash\_code());

json.addObject(L"date", Date2::serialize());

json.addString(L"name", this->name);

json.addInteger(L"length", this->length);

json.addObject(L"studio", this->studio.serialize());

return json;

}

void Disk::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"disk") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

this->length = json.getInteger(L"length");

Date2::deserialize(json.getObject(L"date"));

this->studio.deserialize(json.getObject(L"studio"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Disk::operator=(const Disk& reference) {

this->length = reference.length;

this->studio = reference.studio;

Date2::operator=(reference);

Nameable::operator=(reference);

}

Nameable.h

#pragma once

#include "String.h"

#include "FWString.h"

class Nameable {

protected:

FWString name;

public:

Nameable(const FWString name) : name(name) {}

Nameable() : Nameable(L"") {}

Nameable(const Nameable& reference) {

this->name = reference.name;

}

~Nameable() {}

void setName(const FWString name) {

this->name = name;

}

FWString getName() const {

return this->name;

}

public:

void operator= (const Nameable& reference);

};

Nameable.cpp

#include "Nameable.h"

void Nameable::operator=(const Nameable& reference) {

this->name = reference.name;

}

Operation.h

#pragma once

#include "Returnable.h"

#include "Abonent.h"

#include "Date2.h"

#include "Disk.h"

class Operation : public Returnable, public Fileable, public Serializeable, public Object {

protected:

Date2 date;

Abonent abonent;

Disk disk;

public:

Operation(Date2 date, Abonent abonent, Disk disk, bool isreturn) : Returnable(isreturn), date(date), abonent(abonent), disk(disk) {}

Operation() : Returnable(), date(Date2()), abonent(Abonent()), disk(Disk()) {}

~Operation() {}

Operation(const Operation & reference);

void setDate(const Date2 date);

Date2 getDate() const;

void setAbonent(const Abonent abonent);

Abonent getAbonent() const;

void setDisk(const Disk disk);

Disk getDisk() const;

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Operation& reference);

};

Operation.cpp

#include "Operation.h"

Operation::Operation(const Operation& reference) {

this->date = reference.date;

this->abonent = reference.abonent;

this->disk = reference.disk;

}

void Operation::setDate(const Date2 date) {

this->date = date;

}

Date2 Operation::getDate() const {

return this->date;

}

void Operation::setAbonent(const Abonent abonent) {

this->abonent = abonent;

}

Abonent Operation::getAbonent() const {

return this->abonent;

}

void Operation::setDisk(const Disk disk) {

this->disk = disk;

}

Disk Operation::getDisk() const {

return this->disk;

}

void Operation::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Operation).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

stream.write((char\*)& this->isreturn, sizeof(bool));

this->date.save(stream);

this->abonent.save(stream);

this->disk.save(stream);

}

void Operation::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Operation).hash\_code()) {

throw WrongInputFileException();

}

stream.read((char\*)& this->isreturn, sizeof(bool));

this->date.load(stream);

this->abonent.load(stream);

this->disk.load(stream);

}

MagicJSON::JsonObject Operation::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"operation");

json.addInteger(L"\_\_hash", typeid(Operation).hash\_code());

json.addObject(L"date", this->date.serialize());

json.addObject(L"abonent", this->abonent.serialize());

json.addObject(L"disk", this->disk.serialize());

json.addInteger(L"is\_return", (long)(this->isreturn));

return json;

}

void Operation::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"operation") {

throw UnexpectedTypeException();

}

this->date.deserialize(json.getObject(L"date"));

this->abonent.deserialize(json.getObject(L"abonent"));

this->disk.deserialize(json.getObject(L"disk"));

this->isreturn = (bool)json.getInteger(L"is\_return");

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Operation::operator=(const Operation& reference) {

this->date = reference.date;

this->abonent = reference.abonent;

this->disk = reference.disk;

}

Returnable.h

#pragma once

class Returnable {

protected:

bool isreturn;

public:

Returnable(const bool isreturn) : isreturn(isreturn) {}

Returnable() : Returnable(false) {}

~Returnable() {}

Returnable(const Returnable& reference);

bool isReturn() const { return this->isreturn; }

void setType(const bool isreturn) { this->isreturn = isreturn; }

public:

void operator= (const Returnable& reference);

};

Returnable.cpp

#include "Returnable.h"

Returnable::Returnable(const Returnable& reference) {

this->isreturn = reference.isreturn;

}

void Returnable::operator=(const Returnable& reference) {

this->isreturn = reference.isreturn;

}

Studio.h

#pragma once

#include "Date2.h"

#include "Nameable.h"

class Studio : public Date2, public Nameable{

public:

Studio(const FWString name, const short year, const short month, const short day) : Date2(year, month, day), Nameable(name) {}

Studio() : Date2(), Nameable(L"") {}

Studio(const Studio& reference);

~Studio() {}

public:

void save(ofstream& stream) const override;

void load(ifstream& stream) override;

MagicJSON::JsonObject serialize() override;

void deserialize(MagicJSON::JsonObject json) override;

public:

void operator= (const Studio& reference);

};

Studio.h

#include "Studio.h"

Studio::Studio(const Studio& reference) : Date2(reference), Nameable(reference){}

void Studio::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(Studio).hash\_code();

stream.write((char\*)& hash, sizeof(hash\_code));

Date2::save(stream);

this->name.save(stream);

}

void Studio::load(ifstream& stream) {

hash\_code hash = 0;

stream.read((char\*)& hash, sizeof(hash\_code));

if (hash != (hash\_code)typeid(Studio).hash\_code()) {

throw WrongInputFileException();

}

Date2::load(stream);

this->name.load(stream);

}

MagicJSON::JsonObject Studio::serialize() {

MagicJSON::JsonObject json;

json.addString(L"\_\_type", L"studio");

json.addInteger(L"\_\_hash", typeid(Studio).hash\_code());

json.addObject(L"date", Date2::serialize());

json.addString(L"name", this->name);

return json;

}

void Studio::deserialize(MagicJSON::JsonObject json) {

try {

if (json.getString(L"\_\_type") != L"studio") {

throw UnexpectedTypeException();

}

this->name = json.getString(L"name");

Date2::deserialize(json.getObject(L"date"));

}

catch (MagicJSON::NoObjectFoundException e) {

throw IncorrectObjectDataException();

}

}

void Studio::operator=(const Studio& reference) {

Date2::operator=(reference);

Nameable::operator=(reference);

}

Serializeable.h

#pragma once

#include "MagicJSON.h"

#include "SerializeExceptions.h"

class Serializeable {

public:

virtual MagicJSON::JsonObject serialize() = 0;

virtual void deserialize(MagicJSON::JsonObject json) = 0;

};

Saveable.h

#pragma once

#include <fstream>

using namespace std;

class Saveable {

public:

Saveable() {}

~Saveable() {}

virtual void save(ofstream& stream) const {}

};

Loadable.h

#pragma once

#include <fstream>

using namespace std;

class Loadable {

public:

Loadable() {}

~Loadable() {}

virtual void load(ifstream& stream) {}

};

Fileable.h

#pragma once

#include "Loadable.h"

#include "Saveable.h"

typedef long hash\_code;

class Fileable : public Loadable, public Saveable {

public:

Fileable() {}

~Fileable() {}

};

FVector.h

#pragma once

#include "Fileable.h"

#include "HashSet.h"

#include "Object.h"

#include "Exception.h"

#include <vector>

class FVector : public vector<Fileable\*>, public Fileable, public Object{

HashSet<Fileable\* (\*)(void)> creators;

public:

FVector(): vector<Fileable\*>(), Fileable(), Object() {}

FVector(const FVector& reference);

~FVector() {}

virtual Fileable\* get(size\_t index) const;

virtual void save(ofstream& stream) const override;

virtual void load(ifstream& stream) override;

virtual void addObjectCreator(size\_t hash, Fileable\* (\*creator)(void));

virtual Fileable\* (\*getObjectCreator(size\_t hash))(void);

public:

void operator=(const FVector& reference);

};

FVector.cpp

#include "FVector.h"

FVector::FVector(const FVector& reference) : vector<Fileable\*>(reference), Fileable(reference), Object(reference) {

this->creators = reference.creators;

}

Fileable\* FVector::get(size\_t index) const {

if (index < 0 || index >= this->size()) {

throw IndexOutOfBoundsException();

}

return (\*this)[index];

}

void FVector::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(FVector).hash\_code();

stream.write((char\*) & (hash), sizeof(hash\_code));

size\_t quantity = this->size();

stream.write((char\*) & (quantity), sizeof(size\_t));

for (size\_t i = 0; i < quantity; i++) {

this->get(i)->save(stream);

}

}

void FVector::load(ifstream& stream) {

this->clear();

hash\_code hash = 0;

stream.read((char\*) & (hash), sizeof(hash\_code));

if (hash != (hash\_code)typeid(FVector).hash\_code()) {

throw WrongInputFileException();

}

size\_t quantity = 0;

stream.read((char\*) & (quantity), sizeof(size\_t));

for (size\_t i = 0; i < quantity; i++) {

hash\_code hash = 0;

stream.read((char\*) & (hash), sizeof(hash\_code));

Fileable\* object = nullptr;

try {

object = this->creators.get(hash)();

}

catch (NoObjectFoundException e) {

throw UnknownDataTypeException();

}

stream.seekg((-1) \* (long)sizeof(hash\_code), ios::cur);

object->load(stream);

this->push\_back(object);

}

}

void FVector::addObjectCreator(size\_t hash, Fileable\* (\*creator)(void)) {

this->creators.add(hash, creator);

}

Fileable\* (\*FVector::getObjectCreator(size\_t hash))(void) {

return this->creators.get(hash);

}

void FVector::operator=(const FVector& reference) {

vector<Fileable\*>::operator=(reference);

this->creators = reference.creators;

}

FWString.h

#pragma once

#include "Fileable.h"

#include "Exception.h"

#include <windows.h>

class FWString : public wstring, public Fileable {

public:

FWString() : wstring(), Fileable() {}

FWString(const LPCWSTR str) : wstring(str), Fileable() {}

FWString(const FWString& reference) : wstring(reference), Fileable(reference) {}

FWString(const wstring& reference) : wstring(reference) {}

~FWString() {}

virtual void save(ofstream& stream) const override;

virtual void load(ifstream& stream) override;

public:

using wstring::operator+=;

using wstring::operator[];

using wstring::operator=;

};

FWString.cpp

#include "FWString.h"

void FWString::save(ofstream& stream) const {

hash\_code hash = (hash\_code)typeid(FWString).hash\_code();

stream.write((char\*) & (hash), sizeof(hash\_code));

size\_t length = this->length() + 1;

stream.write((char\*) & (length), sizeof(size\_t));

LPCWSTR str = this->c\_str();

stream.write((char\*)str, length\*sizeof(WCHAR));

}

void FWString::load(ifstream& stream) {

this->clear();

hash\_code hash = 0;

stream.read((char\*) & (hash), sizeof(hash\_code));

if (hash != (hash\_code)typeid(FWString).hash\_code()) {

throw WrongInputFileException();

}

size\_t length = 0;

stream.read((char\*) & (length), sizeof(size\_t));

LPCWSTR str = new WCHAR[length];

stream.read((char\*)str, length\*sizeof(WCHAR));

this->append(str);

delete[] str;

}

HashSet.h

#pragma once

#include "Object.h"

#include "Exception.h"

#include <set>

#include <type\_traits>

namespace {

template <class T>

class HashSetNode : public Object {

public:

T value;

size\_t key;

HashSetNode(size\_t key, T value) : value(value), key(key) {}

HashSetNode() : HashSetNode(0, T()) {}

};

}

template <class T>

class HashSet : protected set<HashSetNode<T>, bool(\*)(const HashSetNode<T>& less, const HashSetNode<T>& higher)>, public Object{

typedef set<HashSetNode<T>, bool(\*)(const HashSetNode<T>& less, const HashSetNode<T>& higher)> Set;

public:

HashSet() : Set(&comparator) {}

HashSet(const HashSet& reference) : Set(reference), Object(reference){}

~HashSet() {}

virtual void add(size\_t key, T element);

virtual T get(const size\_t key) const;

using Set::cend;

using Set::cbegin;

using Set::size;

using Set::operator=;

using Set::find;

using Set::empty;

using Set::max\_size;

using Set::clear;

using Set::erase;

protected:

static bool comparator(const HashSetNode<T>& less, const HashSetNode<T>& higher) {

return less.key < higher.key;

}

};

template<class T>

inline void HashSet<T>::add(size\_t key, T element) {

this->insert(HashSetNode<T>(key, element));

}

template<class T>

inline T HashSet<T>::get(const size\_t key) const {

if (!this->count(HashSetNode<T>(key, T()))) {

throw NoObjectFoundException();

}

return this->find(HashSetNode<T>(key, T()))->value;

}

NetworkMessageKeys.h

#pragma once

#include <string>

using namespace std;

// KEYS

const wstring COMMAND\_TYPE\_KEY = L"command\_type";

const wstring DATA\_TYPE\_KEY = L"data\_type";

const wstring REPORT\_TYPE\_KEY = L"report\_type";

const wstring FILTER\_TYPE\_KEY = L"filter\_type";

const wstring FILTER\_VALUE\_KEY = L"filter\_value";

const wstring LOAD\_DATA\_KEY = L"load\_data";

const wstring SAVE\_DATA\_KEY = L"save\_data";

const wstring VALUE\_KEY = L"value";

const wstring SIZE\_KEY = L"size";

const wstring PATH\_KEY = L"path";

const wstring ERROR\_TYPE\_KEY = L"error\_type";

const wstring SUCCESS\_TYPE\_KEY = L"success\_type";

// COMANDS

const wstring COMMAND\_GET\_DATA = L"get\_data";

const wstring COMMAND\_GET\_REPORT = L"get\_report";

const wstring COMMAND\_ADD\_DATA = L"add\_data";

const wstring COMMAND\_SEND\_DATA = L"send\_data";

const wstring COMMAND\_ERROR = L"error";

const wstring COMMAND\_SUCCESS = L"success";

const wstring COMMAND\_LOAD = L"load\_data";

const wstring COMMAND\_SAVE = L"save\_data";

const wstring COMMAND\_TERMINATE = L"terminate";

// ERROR VALUES

const wstring ERROR\_INVALID\_COMMAND = L"invalid\_command";

const wstring ERROR\_INVALID\_FILE = L"invalid\_file";

const wstring ERROR\_INVALID\_VALUES = L"invalid\_values";

// SUCCESS VALUES

const wstring SUCCESS\_READING\_FILE = L"success\_reading\_file";

const wstring SUCCESS\_ADDING\_DATA = L"success\_adding\_data";

// DATA TYPES

const wstring DATA\_ALL = L"data\_all";

const wstring DATA\_FILTERED = L"data\_fitered";

const wstring DATA\_OPERATION = L"data\_operation";

// FILTER TYPES

const wstring FILTER\_TYPE\_STRING = L"filter\_string";

const wstring FILTER\_TYPE\_INTEGER = L"filter\_integer";

// LOAD TYPES

const wstring LOAD\_BINARY = L"load\_binary";

const wstring LOAD\_TEXT = L"load\_text";

// SAVE TYPES

const wstring SAVE\_BINARY = L"save\_binary";

const wstring SAVE\_TEXT = L"save\_text";

// REPORT TYPES

const wstring REPORT\_STATISTIC = L"report\_statistic";

NetworkMessagesHandler.h

#pragma once

#include "TurboPipes.h"

#include "DailyReport.h"

#include "NetworkMessageKeys.h"

#include <string>

#include <iostream>

using namespace std;

class NetworkMessagesHandler : public TurboPipes::PipeableString {

protected:

DailyReport\* dailyReport;

public:

NetworkMessagesHandler(DailyReport\* dailyReport);

~NetworkMessagesHandler();

public:

void handleMessage(wstring& message) override;

protected:

void handleGetAllMessage();

void handleGetFilteredMessage(MagicJSON::JsonObject message);

void handleLoadTextFileMessage(MagicJSON::JsonObject message);

void handleSaveTextFileMessage(MagicJSON::JsonObject message);

void handleLoadBinaryFileMessage(MagicJSON::JsonObject message);

void handleSaveBinaryFileMessage(MagicJSON::JsonObject message);

void handleAddOperationMessage(MagicJSON::JsonObject message);

void handleReportStaticticMessage(MagicJSON::JsonObject message);

static MagicJSON::JsonObject buildOperationJson(Operation\* operation);

};

NetworkMessagesHandler.cpp

#include "NetworkMessagesHandler.h"

#include "base\_64.h"

#include "StatTools.h"

#include <locale>

#include <codecvt>

#include <string>

#include <chrono>

#include <ctime>

NetworkMessagesHandler::NetworkMessagesHandler(DailyReport\* dailyReport) : dailyReport(dailyReport) {

if (!this->dailyReport) {

throw ZeroPointerException();

}

}

NetworkMessagesHandler::~NetworkMessagesHandler() {}

void NetworkMessagesHandler::handleMessage(wstring& message) {

try{

std::time\_t current\_time = std::chrono::system\_clock::to\_time\_t(std::chrono::system\_clock::now());

char buff[30];

ctime\_s(buff, 30, &current\_time);

wcout << buff << "recieved: " << message << endl << endl;

MagicJSON::JsonObject json\_message(message);

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_GET\_DATA) == 0) {

if (json\_message.getString(DATA\_TYPE\_KEY).compare(DATA\_ALL) == 0) {

this->handleGetAllMessage();

}

if (json\_message.getString(DATA\_TYPE\_KEY).compare(DATA\_FILTERED) == 0) {

this->handleGetFilteredMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_LOAD) == 0) {

if (json\_message.getString(LOAD\_DATA\_KEY).compare(LOAD\_BINARY) == 0) {

this->handleLoadBinaryFileMessage(json\_message);

}

if (json\_message.getString(LOAD\_DATA\_KEY).compare(LOAD\_TEXT) == 0) {

this->handleLoadTextFileMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_SAVE) == 0) {

if (json\_message.getString(SAVE\_DATA\_KEY).compare(SAVE\_BINARY) == 0) {

this->handleSaveBinaryFileMessage(json\_message);

}

if (json\_message.getString(SAVE\_DATA\_KEY).compare(SAVE\_TEXT) == 0) {

this->handleSaveTextFileMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_ADD\_DATA) == 0) {

if (json\_message.getString(DATA\_TYPE\_KEY).compare(DATA\_OPERATION) == 0) {

this->handleAddOperationMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_GET\_REPORT) == 0) {

if (json\_message.getString(REPORT\_TYPE\_KEY).compare(REPORT\_STATISTIC) == 0) {

this->handleReportStaticticMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_TERMINATE) == 0) {

this->dispatcher->stopThread();

}

}

catch (MagicJSON::JsonException e) {

wcout << "Error: recieved error type message" << endl;

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_ERROR);

answer\_json.addString(ERROR\_TYPE\_KEY, ERROR\_INVALID\_COMMAND);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

}

void NetworkMessagesHandler::handleGetAllMessage() {

MagicJSON::JsonObject dailyReport\_json;

MagicJSON::JsonArray returns;

MagicJSON::JsonArray extraditions;

for (size\_t i = 0; i < this->dailyReport->getReturnsSize(); i++) {

Operation\* operation = this->dailyReport->getReturn(i);

returns.addObject(buildOperationJson(operation));

}

for (size\_t i = 0; i < this->dailyReport->getExtraditionsSize(); i++) {

Operation\* operation = this->dailyReport->getExtradition(i);

extraditions.addObject(buildOperationJson(operation));

}

dailyReport\_json.addArray(L"returns", returns);

dailyReport\_json.addArray(L"extraditions", extraditions);

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SEND\_DATA);

answer\_json.addString(DATA\_TYPE\_KEY, DATA\_ALL);

answer\_json.addObject(VALUE\_KEY, dailyReport\_json);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

void NetworkMessagesHandler::handleGetFilteredMessage(MagicJSON::JsonObject message) {

MagicJSON::JsonObject dailyReport\_json;

MagicJSON::JsonArray returns;

MagicJSON::JsonArray extraditions;

for (size\_t i = 0; i < this->dailyReport->getReturnsSize(); i++) {

Operation\* operation = this->dailyReport->getReturn(i);

if (message.getString(FILTER\_TYPE\_KEY).compare(FILTER\_TYPE\_STRING) == 0) {

if (operation->getAbonent().getName().compare(message.getString(FILTER\_VALUE\_KEY)) == 0) {

returns.addObject(buildOperationJson(operation));

}

}

if (message.getString(FILTER\_TYPE\_KEY).compare(FILTER\_TYPE\_INTEGER) == 0) {

if (operation->getAbonent().getYear() == message.getInteger(FILTER\_VALUE\_KEY)) {

returns.addObject(buildOperationJson(operation));

}

}

}

for (size\_t i = 0; i < this->dailyReport->getExtraditionsSize(); i++) {

Operation\* operation = this->dailyReport->getExtradition(i);

if (message.getString(FILTER\_TYPE\_KEY).compare(FILTER\_TYPE\_STRING)) {

if (operation->getAbonent().getName().compare(message.getString(FILTER\_VALUE\_KEY)) == 0) {

extraditions.addObject(buildOperationJson(operation));

}

}

if (message.getString(FILTER\_TYPE\_KEY).compare(FILTER\_TYPE\_INTEGER) == 0) {

if (operation->getAbonent().getYear() == message.getInteger(FILTER\_VALUE\_KEY)) {

extraditions.addObject(buildOperationJson(operation));

}

}

}

dailyReport\_json.addArray(L"returns", returns);

dailyReport\_json.addArray(L"extraditions", extraditions);

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SEND\_DATA);

answer\_json.addString(DATA\_TYPE\_KEY, DATA\_ALL);

answer\_json.addObject(VALUE\_KEY, dailyReport\_json);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

void NetworkMessagesHandler::handleLoadTextFileMessage(MagicJSON::JsonObject message) {

try {

MagicJSON::JsonObject data = message.getObject(VALUE\_KEY);

this->dailyReport->deserialize(data);

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SUCCESS);

answer\_json.addString(SUCCESS\_TYPE\_KEY, SUCCESS\_READING\_FILE);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

catch (IncorrectObjectDataException){

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_ERROR);

answer\_json.addString(ERROR\_TYPE\_KEY, ERROR\_INVALID\_FILE);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

}

void NetworkMessagesHandler::handleSaveTextFileMessage(MagicJSON::JsonObject message) {

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SAVE);

answer\_json.addString(SAVE\_DATA\_KEY, SAVE\_TEXT);

answer\_json.addString(PATH\_KEY, message.getString(PATH\_KEY));

answer\_json.addObject(VALUE\_KEY, this->dailyReport->serialize());

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

void NetworkMessagesHandler::handleLoadBinaryFileMessage(MagicJSON::JsonObject message) {

wstring\_convert<codecvt\_utf8\_utf16<wchar\_t>> converter;

string sbuffer = converter.to\_bytes(message.getString(VALUE\_KEY));

size\_t size = message.getInteger(SIZE\_KEY);

byte\* buffer = new byte[size];

memcpy(buffer, base64\_decode(sbuffer).c\_str(), size\*sizeof(byte));

ofstream out("temp.binary", ios::binary);

out.write((char\*)buffer, size \* sizeof(byte));

out.close();

delete[] buffer;

try {

ifstream in;

in.exceptions(ifstream::failbit | ifstream::badbit);

in.open("temp.binary", ios::binary);

this->dailyReport->load(in);

in.close();

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SUCCESS);

answer\_json.addString(SUCCESS\_TYPE\_KEY, SUCCESS\_READING\_FILE);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

catch (WrongInputDataException e) {

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_ERROR);

answer\_json.addString(ERROR\_TYPE\_KEY, ERROR\_INVALID\_FILE);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

catch (wifstream::failure e) {

wcout << "Error: error trying to access to temp.binary file" << endl;

}

}

void NetworkMessagesHandler::handleSaveBinaryFileMessage(MagicJSON::JsonObject message) {

try {

ofstream out;

out.exceptions(ifstream::failbit | ifstream::badbit);

out.open("temp.binary", ios::binary);

this->dailyReport->save(out);

out.close();

ifstream in;

in.exceptions(ifstream::failbit | ifstream::badbit);

in.open("temp.binary", ios::binary);

in.seekg(0, in.end);

size\_t size = in.tellg();

in.seekg(0, in.beg);

byte\* buffer = new byte[size];

in.read((char\*)buffer, size \* sizeof(byte));

in.close();

string base\_string = base64\_encode(buffer, size \* sizeof(byte));

delete[] buffer;

wstring\_convert<codecvt\_utf8\_utf16<wchar\_t>> converter;

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SAVE);

answer\_json.addString(SAVE\_DATA\_KEY, SAVE\_BINARY);

answer\_json.addInteger(SIZE\_KEY, size);

answer\_json.addString(VALUE\_KEY, converter.from\_bytes(base\_string));

answer\_json.addString(PATH\_KEY, message.getString(PATH\_KEY));

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

catch (wifstream::failure e) {

wcout << "Error: error trying to access to temp.binary file" << endl;

}

}

void NetworkMessagesHandler::handleAddOperationMessage(MagicJSON::JsonObject message) {

MagicJSON::JsonObject json\_operation = message.getObject(VALUE\_KEY);

bool isReturn = false;

if (json\_operation.getString(L"operation\_type").compare(L"return") == 0) {

isReturn = true;

}

long operation\_day = json\_operation.getInteger(L"operation\_day");

long operation\_month = json\_operation.getInteger(L"operation\_month");

long operation\_year = json\_operation.getInteger(L"operation\_year");

wstring abonent\_name = json\_operation.getString(L"abonent\_name");

wstring abonent\_surename = json\_operation.getString(L"abonent\_surename");

long abonent\_month = json\_operation.getInteger(L"abonent\_month");

long abonent\_year = json\_operation.getInteger(L"abonent\_year");

long disk\_lenght = json\_operation.getInteger(L"disk\_length");

wstring disk\_name = json\_operation.getString(L"disk\_name");

long disk\_day = json\_operation.getInteger(L"disk\_day");

long disk\_month = json\_operation.getInteger(L"disk\_month");

long disk\_year = json\_operation.getInteger(L"disk\_year");

wstring studio\_name = json\_operation.getString(L"studio\_name");

long studio\_day = json\_operation.getInteger(L"studio\_day");

long studio\_month = json\_operation.getInteger(L"studio\_month");

long studio\_year = json\_operation.getInteger(L"studio\_year");

try {

Operation\* operation = new Operation(

Date2(operation\_year, operation\_month, operation\_day),

Abonent(abonent\_name, abonent\_surename, abonent\_year, abonent\_month),

Disk(Studio(studio\_name, studio\_year, studio\_month, studio\_day),

disk\_lenght, disk\_name, disk\_year, disk\_month, disk\_day), isReturn);

if (isReturn) {

this->dailyReport->addReturn(operation);

}

else {

this->dailyReport->addExtradition(operation);

}

}

catch (WrongInputValuesException e) {

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_ERROR);

answer\_json.addString(ERROR\_TYPE\_KEY, ERROR\_INVALID\_VALUES);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SUCCESS);

answer\_json.addString(SUCCESS\_TYPE\_KEY, SUCCESS\_ADDING\_DATA);

answer\_json.addObject(VALUE\_KEY, json\_operation);

answer\_json.addString(DATA\_TYPE\_KEY, DATA\_OPERATION);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

void NetworkMessagesHandler::handleReportStaticticMessage(MagicJSON::JsonObject message) {

MagicJSON::JsonObject stat\_json;

stat\_json.addFloat(L"average\_length", StatTools::statistic<int>(this->dailyReport));

stat\_json.addFloat(L"average\_symbols", StatTools::statistic<wstring>(this->dailyReport));

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_GET\_REPORT);

answer\_json.addString(REPORT\_TYPE\_KEY, REPORT\_STATISTIC);

answer\_json.addObject(VALUE\_KEY, stat\_json);

answer\_json.addString(DATA\_TYPE\_KEY, DATA\_OPERATION);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

MagicJSON::JsonObject NetworkMessagesHandler::buildOperationJson(Operation\* operation) {

MagicJSON::JsonObject operation\_json;

operation\_json.addInteger(L"operation\_day", operation->getDate().getDay());

operation\_json.addInteger(L"operation\_month", operation->getDate().getMonth());

operation\_json.addInteger(L"operation\_year", operation->getDate().getYear());

operation\_json.addInteger(L"operation\_is\_return", operation->isReturn());

operation\_json.addString(L"abonent\_name", operation->getAbonent().getName());

operation\_json.addString(L"abonent\_surename", operation->getAbonent().getSurename());

operation\_json.addInteger(L"abonent\_year", operation->getAbonent().getYear());

operation\_json.addInteger(L"abonent\_month", operation->getAbonent().getMonth());

operation\_json.addInteger(L"disk\_length", operation->getDisk().getLength());

operation\_json.addString(L"disk\_name", operation->getDisk().getName());

operation\_json.addInteger(L"disk\_day", operation->getDisk().getDay());

operation\_json.addInteger(L"disk\_month", operation->getDisk().getMonth());

operation\_json.addInteger(L"disk\_year", operation->getDisk().getYear());

operation\_json.addInteger(L"studio\_day", operation->getDisk().getStudio().getDay());

operation\_json.addInteger(L"studio\_month", operation->getDisk().getStudio().getMonth());

operation\_json.addInteger(L"studio\_year", operation->getDisk().getStudio().getYear());

operation\_json.addString(L"studio\_name", operation->getDisk().getStudio().getName());

return operation\_json;

}

Base\_64.h

#pragma once

/\*

base64.cpp and base64.h

Copyright (C) 2004-2008 René Nyffenegger

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

\*/

#include <iostream>

using namespace std;

std::string base64\_encode(unsigned char const\* bytes\_to\_encode, unsigned int in\_len);

std::string base64\_decode(std::string const& encoded\_string);

Base\_54.cpp

/\*

base64.cpp and base64.h

Copyright (C) 2004-2008 René Nyffenegger

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

\*/

#include <iostream>

using namespace std;

static const std::string base64\_chars =

"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

"abcdefghijklmnopqrstuvwxyz"

"0123456789+/";

static inline bool is\_base64(unsigned char c) {

return (isalnum(c) || (c == '+') || (c == '/'));

}

std::string base64\_encode(unsigned char const\* bytes\_to\_encode, unsigned int in\_len) {

std::string ret;

int i = 0;

int j = 0;

unsigned char char\_array\_3[3];

unsigned char char\_array\_4[4];

while (in\_len--) {

char\_array\_3[i++] = \*(bytes\_to\_encode++);

if (i == 3) {

char\_array\_4[0] = (char\_array\_3[0] & 0xfc) >> 2;

char\_array\_4[1] = ((char\_array\_3[0] & 0x03) << 4) + ((char\_array\_3[1] & 0xf0) >> 4);

char\_array\_4[2] = ((char\_array\_3[1] & 0x0f) << 2) + ((char\_array\_3[2] & 0xc0) >> 6);

char\_array\_4[3] = char\_array\_3[2] & 0x3f;

for (i = 0; (i < 4); i++)

ret += base64\_chars[char\_array\_4[i]];

i = 0;

}

}

if (i)

{

for (j = i; j < 3; j++)

char\_array\_3[j] = '\0';

char\_array\_4[0] = (char\_array\_3[0] & 0xfc) >> 2;

char\_array\_4[1] = ((char\_array\_3[0] & 0x03) << 4) + ((char\_array\_3[1] & 0xf0) >> 4);

char\_array\_4[2] = ((char\_array\_3[1] & 0x0f) << 2) + ((char\_array\_3[2] & 0xc0) >> 6);

char\_array\_4[3] = char\_array\_3[2] & 0x3f;

for (j = 0; (j < i + 1); j++)

ret += base64\_chars[char\_array\_4[j]];

while ((i++ < 3))

ret += '=';

}

return ret;

}

std::string base64\_decode(std::string const& encoded\_string) {

int in\_len = encoded\_string.size();

int i = 0;

int j = 0;

int in\_ = 0;

unsigned char char\_array\_4[4], char\_array\_3[3];

std::string ret;

while (in\_len-- && (encoded\_string[in\_] != '=') && is\_base64(encoded\_string[in\_])) {

char\_array\_4[i++] = encoded\_string[in\_]; in\_++;

if (i == 4) {

for (i = 0; i < 4; i++)

char\_array\_4[i] = base64\_chars.find(char\_array\_4[i]);

char\_array\_3[0] = (char\_array\_4[0] << 2) + ((char\_array\_4[1] & 0x30) >> 4);

char\_array\_3[1] = ((char\_array\_4[1] & 0xf) << 4) + ((char\_array\_4[2] & 0x3c) >> 2);

char\_array\_3[2] = ((char\_array\_4[2] & 0x3) << 6) + char\_array\_4[3];

for (i = 0; (i < 3); i++)

ret += char\_array\_3[i];

i = 0;

}

}

if (i) {

for (j = i; j < 4; j++)

char\_array\_4[j] = 0;

for (j = 0; j < 4; j++)

char\_array\_4[j] = base64\_chars.find(char\_array\_4[j]);

char\_array\_3[0] = (char\_array\_4[0] << 2) + ((char\_array\_4[1] & 0x30) >> 4);

char\_array\_3[1] = ((char\_array\_4[1] & 0xf) << 4) + ((char\_array\_4[2] & 0x3c) >> 2);

char\_array\_3[2] = ((char\_array\_4[2] & 0x3) << 6) + char\_array\_4[3];

for (j = 0; (j < i - 1); j++) ret += char\_array\_3[j];

}

return ret;

}

FrontEnd

Menu.h

#pragma once

#include "WMenu.h"

#include "ATable.h"

#include <iostream>

#include <string>

#include <mutex>

using namespace std;

class Menu : public WMenu::WonderMenu {

public:

Menu();

~Menu();

public:

void print() override;

};

Menu.cpp

#include "Menu.h"

#include <codecvt>

#include <locale>

extern mutex console\_writing\_mutex;

Menu::Menu() : WMenu::WonderMenu(L"exit") {}

Menu::~Menu() {}

using namespace ATable;

void Menu::print() {

Table table(DefaultAppearance(), "table");

table.addColumn(new SimpleColumn("Command", 15, "name"));

table.addColumn(new SimpleColumn("Decription", 40, "decription"));

for (WMenu::MenuCommand\* item : \*this) {

std::wstring\_convert<std::codecvt\_utf8\_utf16<wchar\_t>> converter;

table.addCell("name", new StringCell(converter.to\_bytes(item->getKey().c\_str())));

table.addCell("decription", new StringCell(converter.to\_bytes(item->getDescription())));

}

console\_writing\_mutex.lock();

table.print(cout);

console\_writing\_mutex.unlock();

}

MenuCommandAdd.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

const LPCWSTR INCORRECT\_DAY\_MSG = L"Incorrect day, enter integer mote than 0 and less or equal 31";

const LPCWSTR INCORRECT\_MONTH\_MSG = L"Incorrect month, enter integer mote than 0 and less or equal 12";

const LPCWSTR INCORRECT\_YEAR\_MSG = L"Incorrect year, enter integer value";

const LPCWSTR INCORRECT\_NUMBER\_MSG = L"Incorrect number, enter integer value";

const LPCWSTR INCORRECT\_NAME\_MSG = L"Incorrect name, first letter should be capital";

class MenuCommandAdd : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandAdd(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandAdd();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandAdd.cpp

#include "MenuCommandAdd.h"

#include "InputChecker.h"

#include <iterator>

#include <sstream>

#include <vector>

MenuCommandAdd::MenuCommandAdd(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"add", L"Adds an operations to table"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandAdd::~MenuCommandAdd() {}

void MenuCommandAdd::handleCommnad(wstring inputData) {

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

wstring operation\_type;

try {

if (tokens.size() != 3 || tokens[1].compare(L"operation") != 0) {

wcout << "invalid arguments" << endl;

wcout << "usage: add operation [operation\_type]" << endl;

return;

}

if (tokens[2].compare(L"return") == 0) {

operation\_type = L"return";

}

if (tokens[2].compare(L"extradition") == 0) {

operation\_type = L"extradition";

}

if (tokens[2].compare(L"return") != 0 && tokens[2].compare(L"extradition") != 0) {

wcout << "invalid arguments" << endl;

wcout << "usage: add operation [operation\_type]" << endl;

return;

}

}

catch (out\_of\_range e){

wcout << "invalid arguments" << endl;

wcout << "usage: add operation [operation\_type]" << endl;

return;

}

wcout << "input operation day:" << endl;

int operation\_day = stoi(queryInput(&isDay, INCORRECT\_DAY\_MSG));

wcout << "input operation month:" << endl;

int operation\_month = stoi(queryInput(&isMonth, INCORRECT\_MONTH\_MSG));

wcout << "input operation year:" << endl;

int operation\_year = stoi(queryInput(&isInteger, INCORRECT\_YEAR\_MSG));

wcout << "input abonent name:" << endl;

wstring abonent\_name = queryInput(&isName, INCORRECT\_NAME\_MSG);

wcout << "input abonent surename:" << endl;

wstring abonent\_surename = queryInput(&isName, INCORRECT\_NAME\_MSG);

wcout << "input abonent birth month:" << endl;

int abonent\_month = stoi(queryInput(&isMonth, INCORRECT\_MONTH\_MSG));

wcout << "input abonent birth year:" << endl;

int abonent\_year = stoi(queryInput(&isInteger, INCORRECT\_YEAR\_MSG));

wcout << "input disk length:" << endl;

int disk\_length = stoi(queryInput(&isInteger, INCORRECT\_NUMBER\_MSG));

wcout << "input disk name:" << endl;

wstring disk\_name = queryInput(&isName, INCORRECT\_NAME\_MSG);

wcout << "input disk day:" << endl;

int disk\_day = stoi(queryInput(&isDay, INCORRECT\_DAY\_MSG));

wcout << "input disk month:" << endl;

int disk\_month = stoi(queryInput(&isMonth, INCORRECT\_MONTH\_MSG));

wcout << "input disk year:" << endl;

int disk\_year = stoi(queryInput(&isInteger, INCORRECT\_YEAR\_MSG));

wcout << "input studio name:" << endl;

wstring studio\_name = queryInput(&isName, INCORRECT\_NAME\_MSG);

wcout << "input studio day:" << endl;

int studio\_day = stoi(queryInput(&isDay, INCORRECT\_DAY\_MSG));

wcout << "input studio month:" << endl;

int studio\_month = stoi(queryInput(&isMonth, INCORRECT\_MONTH\_MSG));

wcout << "input studio year:" << endl;

int studio\_year = stoi(queryInput(&isInteger, INCORRECT\_NAME\_MSG));

MagicJSON::JsonObject json;

json.addString(L"operation\_type", operation\_type);

json.addInteger(L"operation\_day", operation\_day);

json.addInteger(L"operation\_month", operation\_month);

json.addInteger(L"operation\_year", operation\_year);

json.addString(L"abonent\_name", abonent\_name);

json.addString(L"abonent\_surename", abonent\_surename);

json.addInteger(L"abonent\_month", abonent\_month);

json.addInteger(L"abonent\_year", abonent\_year);

json.addInteger(L"disk\_length", disk\_length);

json.addString(L"disk\_name", disk\_name);

json.addInteger(L"disk\_day", disk\_day);

json.addInteger(L"disk\_month", disk\_month);

json.addInteger(L"disk\_year", disk\_year);

json.addString(L"studio\_name", studio\_name);

json.addInteger(L"studio\_day", studio\_day);

json.addInteger(L"studio\_month", studio\_month);

json.addInteger(L"studio\_year", studio\_year);

MagicJSON::JsonObject json\_message;

json\_message.addString(COMMAND\_TYPE\_KEY, COMMAND\_ADD\_DATA);

json\_message.addString(DATA\_TYPE\_KEY, DATA\_OPERATION);

json\_message.addObject(VALUE\_KEY, json);

wstring string\_message = json\_message.toString();

this->dispatcher->throwMessage(string\_message);

}

MenuCommandExit.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandExit : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandExit(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandExit();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandExit.cpp

#include "MenuCommandExit.h"

#include <iterator>

#include <sstream>

#include <vector>

#include <iostream>

#include <stdexcept>

MenuCommandExit::MenuCommandExit(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"exit", L"Stops the session"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandExit::~MenuCommandExit()

{

}

void MenuCommandExit::handleCommnad(wstring inputData) {

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

if (tokens.size() > 2) {

wcout << "invalid arguments" << endl;

wcout << "usage: exit" << endl;

wcout << "usage: exit -stop\_server" << endl;

return;

}

try {

if (tokens.at(1).compare(L"-stop\_server") == 0) {

MagicJSON::JsonObject message\_json;

message\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_TERMINATE);

wstring string\_message = message\_json.toString();

this->dispatcher->throwMessage(string\_message);

Sleep(100);

}

else {

wcout << "invalid arguments" << endl;

wcout << "usage: exit" << endl;

wcout << "usage: exit -stop\_server" << endl;

return;

}

}

catch (std::out\_of\_range e) {}

this->dispatcher->stopThread();

}

MenuCommandFind.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandFind : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandFind(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandFind();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandFind.cpp

#include "MenuCommandFind.h"

#include <iterator>

#include <sstream>

#include <vector>

MenuCommandFind::MenuCommandFind(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"find", L"Prints a table of finded operations"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandFind::~MenuCommandFind() {}

void MenuCommandFind::handleCommnad(wstring inputData) {

MagicJSON::JsonObject json;

json.addString(COMMAND\_TYPE\_KEY, COMMAND\_GET\_DATA);

json.addString(DATA\_TYPE\_KEY, DATA\_FILTERED);

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

if(tokens.size() == 3){

if (tokens[1].compare(L"integer") == 0) {

json.addString(FILTER\_TYPE\_KEY, FILTER\_TYPE\_INTEGER);

try {

json.addInteger(FILTER\_VALUE\_KEY, stol(tokens[2]));

}

catch (invalid\_argument e) {

wcout << "invalid arguments: \"" << tokens[2] << "\" is not an integer" << endl;

}

}

if (tokens[1].compare(L"string") == 0) {

json.addString(FILTER\_TYPE\_KEY, FILTER\_TYPE\_STRING);

json.addString(FILTER\_VALUE\_KEY, tokens[2]);

}

}

else {

wcout << "invalid arguments" << endl;

wcout << "usage: find [type] [value]" << endl;

return;

}

wstring message = json.toString();

this->dispatcher->throwMessage(message);

}

MenuCommandLoad.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandLoad : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandLoad(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandLoad();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandLoad.cpp

#include "MenuCommandLoad.h"

#include "Directories.h"

#include <iterator>

#include <sstream>

#include <vector>

#include <fstream>

#include "base\_64.h"

#include <locale>

#include <codecvt>

#include <filesystem>

#include <iostream>

namespace fs = std::experimental::filesystem::v1;

MenuCommandLoad::MenuCommandLoad(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"load", L"Loads data from text or binary file"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandLoad::~MenuCommandLoad() {}

void MenuCommandLoad::handleCommnad(wstring inputData) {

MagicJSON::JsonObject json;

json.addString(COMMAND\_TYPE\_KEY, COMMAND\_LOAD);

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

if (tokens.size() == 3) {

if (tokens[1].compare(L"binary") == 0) {

json.addString(LOAD\_DATA\_KEY, LOAD\_BINARY);

try {

ifstream file;

file.exceptions(std::ifstream::failbit | std::ifstream::badbit);

file.open(FILES\_DIRECTORY + tokens[2], ios::binary);

file.seekg(0, file.end);

size\_t fileLength = file.tellg();

json.addInteger(SIZE\_KEY, fileLength);

file.seekg(0, file.beg);

byte\* buffer = new byte[fileLength];

file.read((char\*)buffer, fileLength\*sizeof(byte));

string base\_64\_buffer = base64\_encode(buffer, fileLength \* sizeof(byte));

wstring\_convert<codecvt\_utf8\_utf16<wchar\_t>> converter;

json.addString(VALUE\_KEY,converter.from\_bytes(base\_64\_buffer));

delete[] buffer;

}

catch (wifstream::failure e) {

wcout << "Error: error reading file" << endl;

wcout << "Available files: " << endl;

//show all files in directory

for (const auto& entry : fs::directory\_iterator(FILES\_DIRECTORY)) {

if (entry.path().extension().compare(BINARY\_EXTENSION) == 0) {

wcout << entry.path().filename() << endl;

}

}

return;

}

}

if (tokens[1].compare(L"text") == 0) {

json.addString(LOAD\_DATA\_KEY, LOAD\_TEXT);

try {

wifstream file;

file.exceptions(std::wifstream::failbit | std::wifstream::badbit);

file.open(FILES\_DIRECTORY + tokens[2]);

wstring buffer;

wstring token;

while (!file.eof()) {

file >> token;

buffer.append(token);

}

json.addObject(VALUE\_KEY, MagicJSON::JsonObject(buffer));

}

catch (wifstream::failure e) {

wcout << "Available files: " << endl;

//show all files in directory

for (const auto& entry : fs::directory\_iterator(FILES\_DIRECTORY)) {

if (entry.path().extension().compare(TEXT\_EXTENSION) == 0) {

wcout << entry.path().filename() << endl;

}

}

wcout << "Error: error reading file" << endl;

return;

}

catch (MagicJSON::ErrorReadingTextException e) {

wcout << "Error: invalid file data" << endl;

return;

}

}

if (tokens[1].compare(L"binary") != 0 && tokens[1].compare(L"text") != 0) {

wcout << "invalid arguments: invalid file type" << endl;

wcout << "usage: load [text/binary] [filepath]" << endl;

return;

}

}

else {

wcout << "invalid arguments" << endl;

wcout << "usage: load [text/binary] [filepath]" << endl;

return;

}

wstring message = json.toString();

this->dispatcher->throwMessage(message);

}

MenuCommandPrint.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

using namespace std;

class MenuCommandPrint : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandPrint(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandPrint();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandPrint.cpp

#include "MenuCommandPrint.h"

MenuCommandPrint::MenuCommandPrint(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"print", L"Prints a table of all operations"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandPrint::~MenuCommandPrint() {}

void MenuCommandPrint::handleCommnad(wstring inputData) {

MagicJSON::JsonObject json;

json.addString(COMMAND\_TYPE\_KEY, COMMAND\_GET\_DATA);

json.addString(DATA\_TYPE\_KEY, DATA\_ALL);

wstring message = json.toString();

this->dispatcher->throwMessage(message);

}

MenuCommandSave.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandSave : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandSave(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandSave();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandSave.cpp

#include "MenuCommandSave.h"

#include <iterator>

#include <sstream>

#include <vector>

#include <fstream>

MenuCommandSave::MenuCommandSave(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"save", L"Saves data to text or binary file"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandSave::~MenuCommandSave() {}

void MenuCommandSave::handleCommnad(wstring inputData) {

MagicJSON::JsonObject json;

json.addString(COMMAND\_TYPE\_KEY, COMMAND\_SAVE);

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

if (tokens.size() == 3) {

if (tokens[1].compare(L"binary") == 0) {

json.addString(SAVE\_DATA\_KEY, SAVE\_BINARY);

}

if (tokens[1].compare(L"text") == 0) {

json.addString(SAVE\_DATA\_KEY, SAVE\_TEXT);

}

if (tokens[1].compare(L"binary") != 0 && tokens[1].compare(L"text") != 0) {

wcout << "invalid arguments: invalid file type" << endl;

wcout << "usage: save [text/binary] [filepath]" << endl;

return;

}

json.addString(PATH\_KEY, tokens[2]);

}

else {

wcout << "invalid arguments" << endl;

wcout << "usage: save [text/binary] [filepath]" << endl;

return;

}

wstring message = json.toString();

this->dispatcher->throwMessage(message);

}

MenuCommandStat.h

#pragma once

#include "WMenu.h"

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include <iostream>

#include <string>

class MenuCommandStat : public WMenu::MenuCommand {

protected:

TurboPipes::PipeDispatcherString\* dispatcher;

public:

MenuCommandStat(TurboPipes::PipeDispatcherString\* dispatcher);

~MenuCommandStat();

public:

void handleCommnad(wstring inputData) override;

};

MenuCommandStat.cpp

#include "MenuCommandStat.h"

#include <iterator>

#include <sstream>

#include <vector>

MenuCommandStat::MenuCommandStat(TurboPipes::PipeDispatcherString\* dispatcher) :

WMenu::MenuCommand(L"stat", L"Returns statistic"), dispatcher(dispatcher) {

if (!this->dispatcher) {

throw exception();

}

}

MenuCommandStat::~MenuCommandStat() {}

void MenuCommandStat::handleCommnad(wstring inputData) {

wistringstream iss(inputData);

vector<wstring> tokens((istream\_iterator<wstring, wchar\_t>(iss)), istream\_iterator<wstring, wchar\_t>());

MagicJSON::JsonObject json\_message;

json\_message.addString(COMMAND\_TYPE\_KEY, COMMAND\_GET\_REPORT);

json\_message.addString(REPORT\_TYPE\_KEY, REPORT\_STATISTIC);

wstring string\_message = json\_message.toString();

this->dispatcher->throwMessage(string\_message);

}

Directories.h

#pragma once

#include <string>

const std::wstring FILES\_DIRECTORY = L"files\\";

const std::wstring BINARY\_EXTENSION = L".binary";

const std::wstring TEXT\_EXTENSION = L".json";

InputChecker.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

bool isDay(wstring& input);

bool isMonth(wstring& input);

bool isInteger(wstring& input);

bool isName(wstring& input);

wstring queryInput(bool (\*checker)(wstring& str), wstring message);

InputChecker.cpp

#include "InputChecker.h"

#include <stdexcept>

bool isDay(wstring& input) {

try {

int result = stoi(input);

if (result > 0 && result <= 31) {

return true;

}

return false;

}

catch (invalid\_argument e){

return false;

}

}

bool isMonth(wstring& input) {

try {

int result = stoi(input);

if (result > 0 && result <= 12) {

return true;

}

return false;

}

catch (invalid\_argument e) {

return false;

}

}

bool isInteger(wstring& input) {

try {

stoi(input);

}

catch (invalid\_argument e) {

return false;

}

return true;

}

bool isName(wstring& input) {

if (input.length() >= 1 && iswupper(input[0])) {

return true;

}

return false;

}

wstring queryInput(bool(\*checker)(wstring& str), wstring message) {

wstring input;

getline(wcin, input);

while (!checker(input)) {

wcout << message << endl;

getline(wcin, input);

}

return input;

}

NetworkMessageKeys.h

#pragma once

#include <string>

using namespace std;

// KEYS

const wstring COMMAND\_TYPE\_KEY = L"command\_type";

const wstring DATA\_TYPE\_KEY = L"data\_type";

const wstring REPORT\_TYPE\_KEY = L"report\_type";

const wstring FILTER\_TYPE\_KEY = L"filter\_type";

const wstring FILTER\_VALUE\_KEY = L"filter\_value";

const wstring LOAD\_DATA\_KEY = L"load\_data";

const wstring SAVE\_DATA\_KEY = L"save\_data";

const wstring VALUE\_KEY = L"value";

const wstring SIZE\_KEY = L"size";

const wstring PATH\_KEY = L"path";

const wstring ERROR\_TYPE\_KEY = L"error\_type";

const wstring SUCCESS\_TYPE\_KEY = L"success\_type";

// COMANDS

const wstring COMMAND\_GET\_DATA = L"get\_data";

const wstring COMMAND\_GET\_REPORT = L"get\_report";

const wstring COMMAND\_ADD\_DATA = L"add\_data";

const wstring COMMAND\_SEND\_DATA = L"send\_data";

const wstring COMMAND\_ERROR = L"error";

const wstring COMMAND\_SUCCESS = L"success";

const wstring COMMAND\_LOAD = L"load\_data";

const wstring COMMAND\_SAVE = L"save\_data";

const wstring COMMAND\_TERMINATE = L"terminate";

// ERROR VALUES

const wstring ERROR\_INVALID\_COMMAND = L"invalid\_command";

const wstring ERROR\_INVALID\_FILE = L"invalid\_file";

const wstring ERROR\_INVALID\_VALUES = L"invalid\_values";

// SUCCESS VALUES

const wstring SUCCESS\_READING\_FILE = L"success\_reading\_file";

const wstring SUCCESS\_ADDING\_DATA = L"success\_adding\_data";

// DATA TYPES

const wstring DATA\_ALL = L"data\_all";

const wstring DATA\_FILTERED = L"data\_fitered";

const wstring DATA\_OPERATION = L"data\_operation";

// FILTER TYPES

const wstring FILTER\_TYPE\_STRING = L"filter\_string";

const wstring FILTER\_TYPE\_INTEGER = L"filter\_integer";

// LOAD TYPES

const wstring LOAD\_BINARY = L"load\_binary";

const wstring LOAD\_TEXT = L"load\_text";

// SAVE TYPES

const wstring SAVE\_BINARY = L"save\_binary";

const wstring SAVE\_TEXT = L"save\_text";

// REPORT TYPES

const wstring REPORT\_STATISTIC = L"report\_statistic";

NetworkMessagesHandler.h

#pragma once

#include "TurboPipes.h"

#include "MagicJSON.h"

#include "NetworkMessageKeys.h"

#include "ATable.h"

#include <string>

#include <iostream>

#include <mutex>

using namespace std;

class NetworkMessagesHandler : public TurboPipes::PipeableString {

public:

NetworkMessagesHandler();

~NetworkMessagesHandler();

public:

void handleMessage(wstring& message) override;

protected:

void handleSendAllDataMessage(MagicJSON::JsonObject message);

void handleSaveTextMessage(MagicJSON::JsonObject message);

void handleSaveBinaryMessage(MagicJSON::JsonObject message);

void handleAddDataSuccessMessage(MagicJSON::JsonObject message);

void handleGetStatisticMessage(MagicJSON::JsonObject message);

static void addOperationToTable(MagicJSON::JsonObject operation, ATable::Table\* table);

static ATable::Table\* constructTableForOperations(string name);

};

NetworkMessagesHandler.cpp

#include "NetworkMessagesHandler.h"

#include "Directories.h"

#include "base\_64.h"

#include <locale>

#include <codecvt>

#include <fstream>

mutex console\_writing\_mutex;

NetworkMessagesHandler::NetworkMessagesHandler() {}

NetworkMessagesHandler::~NetworkMessagesHandler() {}

void NetworkMessagesHandler::handleMessage(wstring& message) {

try {

MagicJSON::JsonObject json\_message(message);

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_SEND\_DATA) == 0) {

this->handleSendAllDataMessage(json\_message);

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_SUCCESS) == 0) {

if (json\_message.getString(SUCCESS\_TYPE\_KEY).compare(SUCCESS\_READING\_FILE) == 0) {

console\_writing\_mutex.lock();

wcout << "Successfuly read file" << endl;

console\_writing\_mutex.unlock();

}

if (json\_message.getString(SUCCESS\_TYPE\_KEY).compare(SUCCESS\_ADDING\_DATA) == 0) {

if (json\_message.getString(DATA\_TYPE\_KEY).compare(DATA\_OPERATION) == 0) {

console\_writing\_mutex.lock();

wcout << "Successfuly added a new operation" << endl;

console\_writing\_mutex.unlock();

this->handleAddDataSuccessMessage(json\_message);

}

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_SAVE) == 0) {

if (json\_message.getString(SAVE\_DATA\_KEY).compare(SAVE\_TEXT) == 0) {

this->handleSaveTextMessage(json\_message);

}

if (json\_message.getString(SAVE\_DATA\_KEY).compare(SAVE\_BINARY) == 0) {

this->handleSaveBinaryMessage(json\_message);

}

}

if (json\_message.getString(COMMAND\_TYPE\_KEY).compare(COMMAND\_GET\_REPORT) == 0) {

if (json\_message.getString(REPORT\_TYPE\_KEY).compare(REPORT\_STATISTIC) == 0) {

this->handleGetStatisticMessage(json\_message);

}

}

}

catch (MagicJSON::JsonException e){

console\_writing\_mutex.lock();

wcout << "Error: recieved error type message" << endl;

console\_writing\_mutex.unlock();

MagicJSON::JsonObject answer\_json;

answer\_json.addString(COMMAND\_TYPE\_KEY, COMMAND\_ERROR);

answer\_json.addString(ERROR\_TYPE\_KEY, ERROR\_INVALID\_COMMAND);

wstring answer\_message = answer\_json.toString();

this->dispatcher->throwMessage(answer\_message);

}

}

void NetworkMessagesHandler::handleSendAllDataMessage(MagicJSON::JsonObject message) {

ATable::Table\* table\_extraditions = constructTableForOperations("extraditions");

ATable::Table\* table\_returns = constructTableForOperations("returns");

MagicJSON::JsonArray extraditions = message.getObject(L"value").getArray(L"extraditions");

MagicJSON::JsonArray returns = message.getObject(L"value").getArray(L"returns");

for (size\_t i = 0; i < extraditions.size(); i++) {

addOperationToTable(extraditions.getObject(i), table\_extraditions);

}

for (size\_t i = 0; i < returns.size(); i++) {

addOperationToTable(returns.getObject(i), table\_returns);

}

console\_writing\_mutex.lock();

wcout << "extraditions:" << endl;

table\_extraditions->print(cout);

wcout << "returns:" << endl;

table\_returns->print(cout);

console\_writing\_mutex.unlock();

delete table\_extraditions;

delete table\_returns;

}

void NetworkMessagesHandler::handleSaveTextMessage(MagicJSON::JsonObject message) {

MagicJSON::JsonObject save\_data = message.getObject(VALUE\_KEY);

try {

wofstream file;

file.exceptions(std::wofstream::failbit | std::wofstream::badbit);

file.open(FILES\_DIRECTORY + message.getString(PATH\_KEY) + TEXT\_EXTENSION);

file << save\_data.toString();

}

catch (wofstream::failure e) {

console\_writing\_mutex.lock();

wcout << "Error: unable to save in file: " << message.getString(PATH\_KEY) << endl;

console\_writing\_mutex.unlock();

return;

}

console\_writing\_mutex.lock();

wcout << "successfully saved file" << endl;

console\_writing\_mutex.unlock();

}

void NetworkMessagesHandler::handleSaveBinaryMessage(MagicJSON::JsonObject message) {

wstring\_convert<codecvt\_utf8\_utf16<wchar\_t>> converter;

string base\_string = base64\_decode(converter.to\_bytes(message.getString(VALUE\_KEY)));

size\_t size = message.getInteger(SIZE\_KEY);

ofstream file;

try {

file.exceptions(std::ofstream::failbit | std::ofstream::badbit);

file.open(FILES\_DIRECTORY + message.getString(PATH\_KEY) + BINARY\_EXTENSION, ios::binary);

}

catch (ofstream::failure e) {

console\_writing\_mutex.lock();

wcout << "Error: unable to save in file: " << message.getString(PATH\_KEY) << endl;

console\_writing\_mutex.unlock();

return;

}

byte\* buffer = new byte[size];

memcpy(buffer, base\_string.c\_str(), size\*sizeof(byte));

file.write((char\*)buffer, size\*sizeof(byte));

delete[] buffer;

console\_writing\_mutex.lock();

wcout << "successfully saved file" << endl;

console\_writing\_mutex.unlock();

}

void NetworkMessagesHandler::handleAddDataSuccessMessage(MagicJSON::JsonObject message) {

ATable::Table\* table = constructTableForOperations("new object");

addOperationToTable(message.getObject(VALUE\_KEY), table);

table->print(cout);

delete table;

}

void NetworkMessagesHandler::handleGetStatisticMessage(MagicJSON::JsonObject message) {

ATable::Table\* table = new ATable::Table(ATable::DefaultAppearance(), "statistic");

table->addColumn(new ATable::SimpleColumn("Average films leghth", 30, "film\_length"));

table->addColumn(new ATable::SimpleColumn("Average film names leghth", 30, "name\_length"));

table->addCell("film\_length", new ATable::DoubleCell(message.getObject(VALUE\_KEY).getFloat(L"average\_length")));

table->addCell("name\_length", new ATable::DoubleCell(message.getObject(VALUE\_KEY).getFloat(L"average\_symbols")));

console\_writing\_mutex.lock();

table->print(cout);

console\_writing\_mutex.unlock();

delete table;

}

void NetworkMessagesHandler::addOperationToTable(MagicJSON::JsonObject operation, ATable::Table\* table) {

wstring\_convert<std::codecvt\_utf8\_utf16<wchar\_t>> converter;

table->addCell("operation\_year", new ATable::LongCell(operation.getInteger(L"operation\_year")));

table->addCell("operation\_month", new ATable::LongCell(operation.getInteger(L"operation\_month")));

table->addCell("operation\_day", new ATable::LongCell(operation.getInteger(L"operation\_day")));

table->addCell("abonent\_name", new ATable::StringCell(converter.to\_bytes(operation.getString(L"abonent\_name"))));

table->addCell("abonent\_surename", new ATable::StringCell(converter.to\_bytes(operation.getString(L"abonent\_surename"))));

table->addCell("abonent\_year", new ATable::LongCell(operation.getInteger(L"abonent\_year")));

table->addCell("abonent\_month", new ATable::LongCell(operation.getInteger(L"abonent\_month")));

table->addCell("disk\_length", new ATable::LongCell(operation.getInteger(L"disk\_length")));

table->addCell("disk\_name", new ATable::StringCell(converter.to\_bytes(operation.getString(L"disk\_name"))));

table->addCell("disk\_year", new ATable::LongCell(operation.getInteger(L"disk\_year")));

table->addCell("disk\_month", new ATable::LongCell(operation.getInteger(L"disk\_month")));

table->addCell("disk\_day", new ATable::LongCell(operation.getInteger(L"disk\_day")));

table->addCell("studio\_name", new ATable::StringCell(converter.to\_bytes(operation.getString(L"studio\_name"))));

table->addCell("studio\_year", new ATable::LongCell(operation.getInteger(L"studio\_year")));

table->addCell("studio\_month", new ATable::LongCell(operation.getInteger(L"studio\_month")));

table->addCell("studio\_day", new ATable::LongCell(operation.getInteger(L"studio\_day")));

}

ATable::Table\* NetworkMessagesHandler::constructTableForOperations(string name) {

ATable::Table\* table = new ATable::Table(ATable::DefaultAppearance(), name);

table->addColumn(new ATable::SimpleColumn("OpY", 4, "operation\_year"));

table->addColumn(new ATable::SimpleColumn("OpM", 3, "operation\_month"));

table->addColumn(new ATable::SimpleColumn("OpD", 3, "operation\_day"));

table->addColumn(new ATable::ConstColumn("->", "separator\_1"));

table->addColumn(new ATable::SimpleColumn("Abon name", 10, "abonent\_name"));

table->addColumn(new ATable::SimpleColumn("Abon sname", 10, "abonent\_surename"));

table->addColumn(new ATable::SimpleColumn("AbY", 10, "abonent\_year"));

table->addColumn(new ATable::SimpleColumn("AbM", 10, "abonent\_month"));

table->addColumn(new ATable::SimpleColumn("Disk Len", 8, "disk\_length"));

table->addColumn(new ATable::SimpleColumn("Disk", 10, "disk\_name"));

table->addColumn(new ATable::SimpleColumn("DsY", 4, "disk\_year"));

table->addColumn(new ATable::SimpleColumn("DsM", 3, "disk\_month"));

table->addColumn(new ATable::SimpleColumn("DsD", 3, "disk\_day"));

table->addColumn(new ATable::SimpleColumn("Studio", 10, "studio\_name"));

table->addColumn(new ATable::SimpleColumn("StY", 4, "studio\_year"));

table->addColumn(new ATable::SimpleColumn("StM", 3, "studio\_month"));

table->addColumn(new ATable::SimpleColumn("StD", 3, "studio\_day"));

return table;

}

base\_64.h

#pragma once

/\*

base64.cpp and base64.h

Copyright (C) 2004-2008 René Nyffenegger

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

\*/

#include <iostream>

using namespace std;

std::string base64\_encode(unsigned char const\* bytes\_to\_encode, unsigned int in\_len);

std::string base64\_decode(std::string const& encoded\_string);

base\_64.cpp

/\*

base64.cpp and base64.h

Copyright (C) 2004-2008 René Nyffenegger

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

\*/

#include <iostream>

using namespace std;

static const std::string base64\_chars =

"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

"abcdefghijklmnopqrstuvwxyz"

"0123456789+/";

static inline bool is\_base64(unsigned char c) {

return (isalnum(c) || (c == '+') || (c == '/'));

}

std::string base64\_encode(unsigned char const\* bytes\_to\_encode, unsigned int in\_len) {

std::string ret;

int i = 0;

int j = 0;

unsigned char char\_array\_3[3];

unsigned char char\_array\_4[4];

while (in\_len--) {

char\_array\_3[i++] = \*(bytes\_to\_encode++);

if (i == 3) {

char\_array\_4[0] = (char\_array\_3[0] & 0xfc) >> 2;

char\_array\_4[1] = ((char\_array\_3[0] & 0x03) << 4) + ((char\_array\_3[1] & 0xf0) >> 4);

char\_array\_4[2] = ((char\_array\_3[1] & 0x0f) << 2) + ((char\_array\_3[2] & 0xc0) >> 6);

char\_array\_4[3] = char\_array\_3[2] & 0x3f;

for (i = 0; (i < 4); i++)

ret += base64\_chars[char\_array\_4[i]];

i = 0;

}

}

if (i)

{

for (j = i; j < 3; j++)

char\_array\_3[j] = '\0';

char\_array\_4[0] = (char\_array\_3[0] & 0xfc) >> 2;

char\_array\_4[1] = ((char\_array\_3[0] & 0x03) << 4) + ((char\_array\_3[1] & 0xf0) >> 4);

char\_array\_4[2] = ((char\_array\_3[1] & 0x0f) << 2) + ((char\_array\_3[2] & 0xc0) >> 6);

char\_array\_4[3] = char\_array\_3[2] & 0x3f;

for (j = 0; (j < i + 1); j++)

ret += base64\_chars[char\_array\_4[j]];

while ((i++ < 3))

ret += '=';

}

return ret;

}

std::string base64\_decode(std::string const& encoded\_string) {

int in\_len = encoded\_string.size();

int i = 0;

int j = 0;

int in\_ = 0;

unsigned char char\_array\_4[4], char\_array\_3[3];

std::string ret;

while (in\_len-- && (encoded\_string[in\_] != '=') && is\_base64(encoded\_string[in\_])) {

char\_array\_4[i++] = encoded\_string[in\_]; in\_++;

if (i == 4) {

for (i = 0; i < 4; i++)

char\_array\_4[i] = base64\_chars.find(char\_array\_4[i]);

char\_array\_3[0] = (char\_array\_4[0] << 2) + ((char\_array\_4[1] & 0x30) >> 4);

char\_array\_3[1] = ((char\_array\_4[1] & 0xf) << 4) + ((char\_array\_4[2] & 0x3c) >> 2);

char\_array\_3[2] = ((char\_array\_4[2] & 0x3) << 6) + char\_array\_4[3];

for (i = 0; (i < 3); i++)

ret += char\_array\_3[i];

i = 0;

}

}

if (i) {

for (j = i; j < 4; j++)

char\_array\_4[j] = 0;

for (j = 0; j < 4; j++)

char\_array\_4[j] = base64\_chars.find(char\_array\_4[j]);

char\_array\_3[0] = (char\_array\_4[0] << 2) + ((char\_array\_4[1] & 0x30) >> 4);

char\_array\_3[1] = ((char\_array\_4[1] & 0xf) << 4) + ((char\_array\_4[2] & 0x3c) >> 2);

char\_array\_3[2] = ((char\_array\_4[2] & 0x3) << 6) + char\_array\_4[3];

for (j = 0; (j < i - 1); j++) ret += char\_array\_3[j];

}

return ret;

}

OOP\_Course\_v001\_console\_front.cpp

#include <iostream>

#include "MagicJSON.h"

#include "ATable.h"

#include "Menu.h"

#include "MenuCommandPrint.h"

#include "MenuCommandFind.h"

#include "MenuCommandLoad.h"

#include "MenuCommandSave.h"

#include "MenuCommandAdd.h"

#include "MenuCommandStat.h"

#include "MenuCommandExit.h"

#include "TurboPipes.h"

#include "NetworkMessagesHandler.h"

#include <vector>

#include <locale>

#include <codecvt>

#include <string>

using namespace std;

const LPCWSTR PIPE\_NAME = L"\\pipe\\$MyPipe$";

wstring PIPE\_ADRESS;

int main(int argc, char\* argv[]) {

wcout << "Copyright (C) Andrieiev Danil 2019" << endl;

wcout << "Vortex Client startup" << endl;

// checking arguments

vector<string> args;

for (int i = argc - 1; i > 0; i--) {

args.push\_back(string(argv[i]));

}

if (args.size() > 1) {

wcout << "Invalid arguments: " << endl << "usage: program.exe [server ip or localhost]" << endl;

return -1;

}

if (args.size() == 1) {

if (args[0].compare("localhost")) {

PIPE\_ADRESS = L"\\\\.";

PIPE\_ADRESS.append(PIPE\_NAME);

}

else {

wstring\_convert<codecvt\_utf8\_utf16<wchar\_t>> converter;

PIPE\_ADRESS.append(L"\\\\");

PIPE\_ADRESS = converter.from\_bytes(args[0]);

PIPE\_ADRESS.append(PIPE\_NAME);

}

}

else {

PIPE\_ADRESS = L"\\\\.";

PIPE\_ADRESS.append(PIPE\_NAME);

}

// creating network messages handler

NetworkMessagesHandler p;

wcout << "Client attempting to connect to server" << endl;

// creating network dispatcher

TurboPipes::PipeDispatcherString dispatcher(PIPE\_ADRESS, false, &p);

wcout << "Client has been connected" << endl;

// creating menu

Menu menu;

// adding commands menu

menu.addCommand(new MenuCommandPrint(&dispatcher));

menu.addCommand(new MenuCommandFind(&dispatcher));

menu.addCommand(new MenuCommandLoad(&dispatcher));

menu.addCommand(new MenuCommandSave(&dispatcher));

menu.addCommand(new MenuCommandAdd(&dispatcher));

menu.addCommand(new MenuCommandStat(&dispatcher));

menu.addCommand(new MenuCommandExit(&dispatcher));

// starting menu loop

menu.start();

}

MagicJSON library

JsonArray.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonItem.h"

#include "JsonObject.h"

#include <vector>

#include <string>

#include "JsonTokenizer.h"

#include "JsonExceptions.h"

namespace MagicJSON {

class JsonArray : protected vector<JsonItem\*>, public JsonItem {

public:

JsonArray() {} // constructor by default

JsonArray(wstring data); // construct json array from string

JsonArray(const JsonArray& reference); // copy constructor

~JsonArray();

public:

virtual JsonArray getArray(size\_t index); // returns json array from index position or throws IndexOutOfRangeException or IncorrectRequestException

virtual double getFloat(size\_t index); // returns floating point number from index position or throws IndexOutOfRangeException or IncorrectRequestException

virtual long getInteger(size\_t index); // returns integer number from index position or throws IndexOutOfRangeException or IncorrectRequestException

virtual wstring getString(size\_t index); // returns string from index position or throws IndexOutOfRangeException or IncorrectRequestException

virtual JsonObject getObject(size\_t index); // returns json object from index position or throws IndexOutOfRangeException or IncorrectRequestException

virtual void addArray(JsonArray value); // adds json array to array

virtual void addFloat(double value); // adds floating point number to array

virtual void addInteger(long value); // adds integer number array to array

virtual void addString(wstring value); // adds string to array

virtual void addObject(JsonObject value); // adds json object to array

public:

virtual wstring toString() override; // returns json array converted to string

virtual JsonItem\* copy() override;

void remove(size\_t index); // removes item on index position from json array or throws IndexOutOfRangeException

public:

using vector<JsonItem\*>::clear;

using vector<JsonItem\*>::capacity;

using vector<JsonItem\*>::empty;

using vector<JsonItem\*>::size;

protected:

void parse(wstring json);

public:

using vector<JsonItem\*>::operator=;

};

}

JsonArray.cpp

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonArray.h"

namespace MagicJSON {

JsonArray::JsonArray(wstring data) {

this->parse(data);

}

JsonArray::JsonArray(const JsonArray& reference) : vector<JsonItem\*>(), JsonItem(reference) {

for (JsonItem\* item : reference) {

this->push\_back(item->copy());

}

}

JsonArray::~JsonArray() {

for (JsonItem\* item : \*this) {

delete item;

}

}

JsonArray JsonArray::getArray(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

if (!dynamic\_cast<JsonArray\*>(this->operator[](index))) {

throw IncorrectRequestException();

}

return \*(dynamic\_cast<JsonArray\*>(this->operator[](index)));

}

double JsonArray::getFloat(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

if (!dynamic\_cast<JsonString\*>(this->operator[](index))) {

throw IncorrectRequestException();

}

try {

return stod(\*(dynamic\_cast<JsonString\*>(this->operator[](index))));

}

catch (invalid\_argument e) {

throw IncorrectRequestException();

}

}

long JsonArray::getInteger(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

if (!dynamic\_cast<JsonString\*>(this->operator[](index))) {

throw IncorrectRequestException();

}

try {

return stol(\*(dynamic\_cast<JsonString\*>(this->operator[](index))));

}

catch (invalid\_argument e) {

throw IncorrectRequestException();

}

}

wstring JsonArray::getString(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

if (!dynamic\_cast<JsonString\*>(this->operator[](index))) {

throw IncorrectRequestException();

}

return \*(dynamic\_cast<JsonString\*>(this->operator[](index)));

}

JsonObject JsonArray::getObject(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

if (!dynamic\_cast<JsonObject\*>(this->operator[](index))) {

throw IncorrectRequestException();

}

return \*(dynamic\_cast<JsonObject\*>(this->operator[](index)));

}

void JsonArray::addArray(JsonArray value) {

this->push\_back(new JsonArray(value));

}

void JsonArray::addFloat(double value) {

this->push\_back(new JsonString(to\_wstring(value)));

}

void JsonArray::addInteger(long value) {

this->push\_back(new JsonString(to\_wstring(value)));

}

void JsonArray::addString(wstring value) {

this->push\_back(new JsonString(value));

}

void JsonArray::addObject(JsonObject value) {

this->push\_back(new JsonObject(value));

}

wstring MagicJSON::JsonArray::toString() {

size\_t number = 0;

wstring json(L"[");

for (JsonItem\* item : \*this) {

json.append(item->toString());

if (number++ != this->size() - 1) {

json.append(L",");

}

}

json.append(L"]");

return json;

}

JsonItem\* JsonArray::copy() {

JsonArray\* arr = new JsonArray();

for (JsonItem\* item : \*this) {

arr->push\_back(item->copy());

}

return arr;

}

void JsonArray::remove(size\_t index) {

if (index >= this->size()) {

throw IndexOutOfRangeException();

}

delete (\*this)[index];

this->erase(this->cbegin() + index);

}

void JsonArray::parse(wstring json) {

JsonTokenizer tokenizer(json);

Token token = tokenizer.next\_token();

while (token.first != JsonTokenizer::EOF\_TOKEN) {

switch (token.first) {

case JsonTokenizer::OBJECT\_TOKEN:

this->addObject(JsonObject(token.second));

break;

case JsonTokenizer::ARRAY\_TOKEN:

this->addArray(JsonArray(token.second));

break;

case JsonTokenizer::INTEGER\_TOKEN:

this->addInteger(stol(token.second));

break;

case JsonTokenizer::FLOAT\_TOKEN:

this->addFloat(stod(token.second));

break;

case JsonTokenizer::STRING\_TOKEN:

this->addString(token.second);

break;

case JsonTokenizer::NULL\_TOKEN:

break;

default:

throw exception();

}

token = tokenizer.next\_token();

if (token.first == JsonTokenizer::COMA\_TOKEN) {

token = tokenizer.next\_token();

}

}

}

}

JsonExceptions.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include <exception>

namespace MagicJSON {

using namespace std;

class JsonException : public exception {

public:

char const\* what() const noexcept override{ return "Error: JsonException"; }

};

class IncorrectRequestException : public JsonException {

public:

virtual char const\* what() const noexcept override { return "Error: IncorrectRequestException"; }

};

class IndexOutOfRangeException : public JsonException {

public:

virtual char const\* what() const noexcept override { return "Error: IndexOutOfRangeException"; }

};

class ReadingException : public JsonException {

public:

virtual char const\* what() const noexcept override { return "Error: ReadingException"; }

};

class NoObjectFoundException : public JsonException {

public:

virtual char const\* what() const noexcept override { return "Error: NoObjectFoundException"; }

};

class ErrorReadingFileException : public ReadingException {

public:

virtual char const\* what() const noexcept override { return "Error: ErrorReadingFileException"; }

};

class ErrorReadingTextException : public ReadingException {

public:

virtual char const\* what() const noexcept override { return "Error: ErrorReadingTextException"; }

};

class UnexpectedFileEndException : public ErrorReadingFileException {

public:

virtual char const\* what() const noexcept override { return "Error: UnexpectedFileEndException"; }

};

class UnexpectedTextEndException : public ErrorReadingTextException {

public:

virtual char const\* what() const noexcept override { return "Error: UnexpectedTextEndException"; }

};

}

JsonItem.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include <string>

namespace MagicJSON {

using namespace std;

class JsonItem {

public:

JsonItem() {}

~JsonItem() {}

virtual wstring toString() = 0;

public:

virtual JsonItem\* copy() = 0;

};

}

JsonObject.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonTokenizer.h"

#include "JsonString.h"

#include <iostream>

#include <set>

#include <utility>

#include <windows.h>

namespace MagicJSON {

using namespace std;

class JsonArray;

typedef pair<wstring, JsonItem\*> JItem;

typedef set<JItem, bool(\*)(JItem, JItem)> JSet;

class JsonObject : protected JSet, public JsonItem {

public:

JsonObject() : JSet(&comparator) {} // constructor by default

JsonObject(wstring data); // construct json hierarchy from string

JsonObject(const JsonObject& reference); // copy constructor

~JsonObject();

public:

virtual void addString(wstring key, wstring value); // adds string field to json

virtual void addArray(wstring key, JsonArray value); // adds json array field to json

virtual void addInteger(wstring key, long value); // adds integer number field to json

virtual void addFloat(wstring key, double value); // adds floating point number field to json

virtual void addObject(wstring key, JsonObject value); // adds another json object field to json

virtual wstring getString(wstring key); // returns string or throws NoObjectFoundException or IncorrectRequestException

virtual JsonArray getArray(wstring key); // returns json array or throws NoObjectFoundException or IncorrectRequestException

virtual long getInteger(wstring key); // returns integer number or throws NoObjectFoundException or IncorrectRequestException

virtual double getFloat(wstring key); // returns floating point number or throws NoObjectFoundException or IncorrectRequestException

virtual JsonObject getObject(wstring key); // returns json object or throws NoObjectFoundException or IncorrectRequestException

public:

virtual wstring toString() override; // returns json hierarchy converted to string

virtual JsonItem\* copy() override;

void remove(wstring key); // removes field with selected key from json hierarchy or throws NoObjectFoundException

public:

using JSet::clear;

protected:

virtual void parse(wstring json);

static bool comparator(JItem less, JItem higher);

public:

using JSet::operator=;

};

}

JsonObject.cpp

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonObject.h"

#include "JsonArray.h"

namespace MagicJSON {

wstring JsonObject::toString() {

size\_t number = 0;

wstring json(L"{");

for (JItem item : \*this) {

json.append(L"\"");

json.append(item.first);

json.append(L"\":");

json.append(item.second->toString());

if (number++ != this->size() - 1) {

json.append(L",");

}

}

json.append(L"}");

return json;

}

JsonItem\* JsonObject::copy() {

JsonObject\* object = new JsonObject();

for (JItem item : \*this) {

JItem newitem(item.first, nullptr);

newitem.second = item.second->copy();

object->insert(newitem);

}

return object;

}

void JsonObject::remove(wstring key) {

if (this->count(JItem(key, nullptr)) == 0) {

throw NoObjectFoundException();

}

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

bool JsonObject::comparator(JItem less, JItem higher) {

return less.first < higher.first;

}

JsonObject::JsonObject(wstring data) : JsonObject() {

this->parse(data);

}

JsonObject::JsonObject(const JsonObject& reference) : JSet(&comparator), JsonItem(reference) {

this->clear();

for (JItem item : reference) {

JItem newitem(item.first, nullptr);

newitem.second = item.second->copy();

this->insert(newitem);

}

}

JsonObject::~JsonObject() {

for (pair<wstring, JsonItem\*> item : \*this) {

delete item.second;

}

}

void JsonObject::addString(wstring key, wstring value) {

if (this->count(JItem(key, nullptr)) != 0){

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

this->insert(JItem(key, new JsonString(value)));

}

void JsonObject::addArray(wstring key, JsonArray value) {

if (this->count(JItem(key, nullptr)) != 0) {

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

this->insert(JItem(key, new JsonArray(value)));

}

void JsonObject::addInteger(wstring key, long value) {

if (this->count(JItem(key, nullptr)) != 0) {

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

this->insert(JItem(key, new JsonString(to\_wstring(value))));

}

void JsonObject::addFloat(wstring key, double value) {

if (this->count(JItem(key, nullptr)) != 0) {

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

this->insert(JItem(key, new JsonString(to\_wstring(value))));

}

void JsonObject::addObject(wstring key, JsonObject value) {

if (this->count(JItem(key, nullptr)) != 0) {

delete this->find(JItem(key, nullptr))->second;

this->erase(JItem(key, nullptr));

}

this->insert(JItem(key, new JsonObject(value)));

}

wstring JsonObject::getString(wstring key) {

if (!this->count(JItem(key, nullptr))) {

throw NoObjectFoundException();

}

iterator iter = this->find(JItem(key, nullptr));

if (iter == this->end()) {

throw NoObjectFoundException();

}

JsonItem\* jitem = iter->second;

JsonString\* jstring = dynamic\_cast<JsonString\*>(jitem);

if (jstring) {

return \*jstring;

}

else {

return iter->second->toString();

}

}

JsonArray JsonObject::getArray(wstring key) {

if (!this->count(JItem(key, nullptr))) {

throw NoObjectFoundException();

}

iterator iter = this->find(JItem(key, nullptr));

if (iter == this->end()) {

throw NoObjectFoundException();

}

if (dynamic\_cast<JsonArray\*>(iter->second) == nullptr) {

throw IncorrectRequestException();

}

return \*dynamic\_cast<JsonArray\*>(iter->second);

}

long JsonObject::getInteger(wstring key) {

if (!this->count(JItem(key, nullptr))) {

throw NoObjectFoundException();

}

iterator iter = this->find(JItem(key, nullptr));

if (iter == this->end()) {

throw NoObjectFoundException();

}

try {

return stol(iter->second->toString());

}

catch (invalid\_argument e) {

throw IncorrectRequestException();

}

}

double JsonObject::getFloat(wstring key) {

if (!this->count(JItem(key, nullptr))) {

throw NoObjectFoundException();

}

iterator iter = this->find(JItem(key, nullptr));

if (iter == this->end()) {

throw NoObjectFoundException();

}

try {

return stod(iter->second->toString());

}

catch (invalid\_argument e) {

throw IncorrectRequestException();

}

}

JsonObject JsonObject::getObject(wstring key) {

if (!this->count(JItem(key, nullptr))) {

throw NoObjectFoundException();

}

iterator iter = this->find(JItem(key, nullptr));

if (iter == this->end()) {

throw NoObjectFoundException();

}

if (dynamic\_cast<JsonObject\*>(iter->second) == nullptr) {

throw IncorrectRequestException();

}

return \*dynamic\_cast<JsonObject\*>(iter->second);

}

void JsonObject::parse(wstring json) {

JsonTokenizer tokenizer(json);

Token token = tokenizer.next\_token();

if (token.first == JsonTokenizer::OBJECT\_TOKEN) {

this->parse(token.second);

return;

}

while (token.first != JsonTokenizer::EOF\_TOKEN) {

if (token.first != JsonTokenizer::STRING\_TOKEN) {

throw ErrorReadingTextException();

}

wstring key = token.second;

token = tokenizer.next\_token();

if (token.first == JsonTokenizer::EOF\_TOKEN || token.first != JsonTokenizer::DOUBLEDOTS\_TOKEN) {

throw UnexpectedTextEndException();

}

token = tokenizer.next\_token();

if (token.first == JsonTokenizer::EOF\_TOKEN) {

throw UnexpectedTextEndException();

}

/\*

if (token.first < JsonTokenizer::STRING\_TOKEN && token.first < JsonTokenizer::NULL\_TOKEN) {

throw exception();

}

\*/

switch (token.first) {

case JsonTokenizer::OBJECT\_TOKEN:

this->addObject(key, token.second);

break;

case JsonTokenizer::ARRAY\_TOKEN:

this->addArray(key, token.second);

break;

case JsonTokenizer::INTEGER\_TOKEN:

this->addInteger(key, stol(token.second));

break;

case JsonTokenizer::FLOAT\_TOKEN:

this->addFloat(key, stod(token.second));

break;

case JsonTokenizer::STRING\_TOKEN:

this->addString(key, token.second);

break;

case JsonTokenizer::NULL\_TOKEN:

break;

default:

throw ErrorReadingTextException();

}

token = tokenizer.next\_token();

if (token.first == JsonTokenizer::COMA\_TOKEN) {

token = tokenizer.next\_token();

}

}

}

}

JsonString.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/#include "JsonItem.h"

#include "windows.h"

#include <string>

namespace MagicJSON {

using namespace std;

class JsonString : public wstring, public JsonItem {

public:

JsonString() : std::wstring() {}

JsonString(LPCWSTR str) : wstring(str) {}

JsonString(wstring str) : wstring(str) {}

JsonString(const JsonString& reference) : wstring(reference), JsonItem(reference) {}

~JsonString() {}

public:

virtual wstring toString() override;

virtual JsonItem\* copy() override;

public:

using wstring::operator=;

using wstring::operator+=;

using wstring::compare;

};

}

JsonString.cpp

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonString.h"

#include <stdexcept>

namespace MagicJSON {

wstring MagicJSON::JsonString::toString() {

try {

stod(\*this);

return \*this;

} catch(invalid\_argument e) {

wstring json(L"\"");

json.append(\*this);

json.append(L"\"");

return json;

}

}

JsonItem\* JsonString::copy() {

return new JsonString(\*this);

}

}

JsonTokenizer.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include <string>

#include <utility>

namespace MagicJSON {

using namespace std;

typedef pair<size\_t, wstring> Token;

class JsonTokenizer {

public:

static const size\_t EOF\_TOKEN = 0;

static const size\_t STRING\_TOKEN = 1;

static const size\_t INTEGER\_TOKEN = 2;

static const size\_t FLOAT\_TOKEN = 3;

static const size\_t ARRAY\_TOKEN = 4;

static const size\_t OBJECT\_TOKEN = 5;

static const size\_t NULL\_TOKEN = 6;

static const size\_t DOUBLEDOTS\_TOKEN = 7;

static const size\_t COMA\_TOKEN = 8;

protected:

size\_t position;

wstring text;

public:

JsonTokenizer(wstring text) : text(text), position(0) {}

~JsonTokenizer() {}

Token next\_token();

protected:

static bool is\_skipable(wchar\_t symbol);

};

}

JsonTokenizer.cpp

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonTokenizer.h"

namespace MagicJSON {

Token JsonTokenizer::next\_token() {

//skipping spaces, enters and tabulation

while (this->position < this->text.length() && is\_skipable(this->text[position])) {

this->position++;

}

if (this->position == this->text.length()) {

return Token(EOF\_TOKEN, L"");

}

wstring token\_text;

wchar\_t symbol = this->text[position++];

switch (symbol) {

case L'\"':

{

while (this->text[this->position] != '\"') {

if (this->position + 1 >= this->text.length()) {

throw exception();

}

token\_text += this->text[this->position];

this->position++;

}

this->position++;

return Token(STRING\_TOKEN, token\_text);

}

break;

case L':':

{

return Token(DOUBLEDOTS\_TOKEN, L":");

}

break;

case L',':

{

return Token(COMA\_TOKEN, L",");

}

break;

case L'{':

{

size\_t not\_closed\_braces = 1;

wstring token\_text;

for (this->position; not\_closed\_braces; this->position++) {

if (this->position >= this->text.length()) {

throw exception();

}

if (this->text[this->position] == L'{') {

not\_closed\_braces++;

}

if (this->text[this->position] == L'}') {

not\_closed\_braces--;

}

if (not\_closed\_braces) {

token\_text += this->text[this->position];

}

}

return Token(OBJECT\_TOKEN, token\_text);

}

break;

case L'[':

{

size\_t not\_closed\_brackets = 1;

wstring token\_text;

for (this->position; not\_closed\_brackets; this->position++) {

if (this->position >= this->text.length()) {

throw exception();

}

if (this->text[this->position] == L'[') {

not\_closed\_brackets++;

}

if (this->text[this->position] == L']') {

not\_closed\_brackets--;

}

if (not\_closed\_brackets) {

token\_text += this->text[this->position];

}

}

return Token(ARRAY\_TOKEN, token\_text);

}

break;

default:

{

bool dot\_flag = false;

wstring token\_text;

if (isdigit(symbol) || symbol == L'-') {

token\_text += symbol;

while (this->position < this->text.length() && (

isdigit(this->text[this->position])

|| this->text[this->position] == L'.')

) {

wchar\_t symbol = this->text[this->position];

if (token\_text == L"-" && !isdigit(symbol)) {

throw exception();

}

if (symbol == L'.') {

if (dot\_flag) {

throw exception();

}

dot\_flag = true;

}

token\_text += symbol;

this->position++;

}

}

if (dot\_flag) {

return Token(FLOAT\_TOKEN, token\_text);

}

return Token(INTEGER\_TOKEN, token\_text);

}

break;

}

}

bool JsonTokenizer::is\_skipable(wchar\_t symbol) {

if (symbol == L' ' || symbol == L'\n' || symbol == L'\t') {

return true;

}

return false;

}

}

MagicJSON.h

#pragma once

/\*

JsonArray.cpp, JsonObject.cpp JsonString.cpp, JsonTokenizer.cpp

JsonArray.h JsonExceptions.h JsonItem.h JsonObject.h JsonString.h JsonTokenizer.h MagicJSO.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.4

\*/

#include "JsonObject.h"

#include "JsonArray.h"

#include "JsonExceptions.h"

ATable library

Appearance.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "TableObject.h"

#include <iostream>

namespace ATable {

class Appearance : public TableObject {

public:

char left\_top\_corner;

char right\_top\_corner;

char left\_bottom\_corner;

char right\_bottom\_corner;

char vertical\_line;

char horizontal\_line;

char left\_separator;

char right\_separator;

char top\_separator;

char bottom\_separator;

char center\_separator;

public:

Appearance();

~Appearance();

void print(ostream & stream); // prints the appearance to stream stream

friend ostream& operator<< (ostream& stream, Appearance obj);

friend ostream& operator<< (ostream& stream, Appearance\* obj);

};

}

Appearance.cpp

#include "Appearance.h"

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

namespace ATable {

Appearance::Appearance() {

this->left\_top\_corner = ' ';

this->right\_top\_corner = ' ';

this->left\_bottom\_corner = ' ';

this->right\_bottom\_corner = ' ';

this->vertical\_line = ' ';

this->horizontal\_line = ' ';

this->left\_separator = ' ';

this->right\_separator = ' ';

this->top\_separator = ' ';

this->bottom\_separator = ' ';

this->center\_separator = ' ';

}

Appearance::~Appearance() {

}

void Appearance::print(ostream& stream) {

stream << "left\_top\_corner: " << left\_top\_corner << endl;

stream << "right\_top\_corner: " << right\_top\_corner << endl;

stream << "left\_bottom\_corner: " << left\_bottom\_corner << endl;

stream << "right\_bottom\_corner: " << right\_bottom\_corner << endl;

stream << "vertical\_line: " << vertical\_line << endl;

stream << "horizontal\_line: " << horizontal\_line << endl;

stream << "left\_separator: " << left\_separator << endl;

stream << "right\_separator: " << right\_separator << endl;

stream << "top\_separator: " << top\_separator << endl;

stream << "bottom\_separator: " << bottom\_separator << endl;

stream << "center\_separator: " << center\_separator << endl;

}

ostream& operator<<(ostream& stream, Appearance obj) {

obj.print(stream);

return stream;

}

ostream& operator<<(ostream& stream, Appearance\* obj) {

return operator<<(stream, \*obj);

}

}

ATable.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Table.h"

#include "SimpleColumn.h"

#include "ConstColumn.h"

#include "IdColumn.h"

#include "DefaultCells.h"

#include "DefaultAppearance.h"

Cell.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "TableObject.h"

#include <string>

#include <iostream>

namespace ATable {

class Cell : public TableObject {

private:

unsigned int width;

public:

Cell();

~Cell();

Cell(Cell\* input);

virtual string output() = 0; // virtual function, which should return string, which well display in table cell

void print(ostream& stream); // prints sata to stream (for debug)

void setWidth(unsigned int width); // sets the width of cell

unsigned int getWidth(); // returns the width of cell

private:

string boundary(string value); // crops the text by cell width

};

}

Cell.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Cell.h"

namespace ATable {

Cell::Cell() {

this->width = 0;

}

Cell::~Cell() {

}

Cell::Cell(Cell \* input) {

this->width = input->width;

}

void Cell::print(ostream& stream) {

stream << this->boundary(this->output());

}

void Cell::setWidth(unsigned int width) {

this->width = width;

}

unsigned int Cell::getWidth() {

return this->width;

}

string Cell::boundary(string value) {

if (value.length() < this->width) {

return value + string(this->width - value.length(), ' ');

}

if (value.length() > this->width) {

string temp = value;

temp.resize(this->width);

return temp;

}

return value;

}

}

Column.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "TableObject.h"

#include "Cell.h"

#include "TableExceptions.h"

#include "Keyable.h"

namespace ATable {

class Column : public TableObject, public Keyable{

public:

unsigned int width;

public:

Column(unsigned int width, string key);

~Column();

unsigned int getWidth(); // returns width of column

void setWidth(unsigned int width); // sets the width of column

virtual Cell\* getCell(int id) = 0; // virtual function, which should return cell on position id

virtual Cell\* getCellClone(int id) = 0; // virtual function, which should return clone of cell on position id

};

}

Column.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Column.h"

namespace ATable {

Column::Column(unsigned int width, string key) : Keyable(key) {

this->width = width;

}

Column::~Column() {

}

unsigned int Column::getWidth() {

return this->width;

}

void Column::setWidth(unsigned int width) {

this->width = width;

}

}

ConstColumn.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Column.h"

#include "DefaultCells.h"

namespace ATable {

class ConstColumn : public Column {

protected:

Cell\* cell;

public:

ConstColumn(string value, string key);

~ConstColumn();

Cell\* getCell(int id); // returns cell cell (id not used)

Cell\* getCellClone(int id); // // returns clone of cell cell (id not used)

};

}

ConstColumn.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "ConstColumn.h"

namespace ATable {

ConstColumn::ConstColumn(string value, string key) : Column(value.length(), key){

this->cell = new StringCell(value);

this->cell->setWidth(this->getWidth());

}

ConstColumn::~ConstColumn() {

delete cell;

}

Cell\* ConstColumn::getCell(int id) {

return this->cell;

}

Cell \* ConstColumn::getCellClone(int id) {

return dynamic\_cast<Cell\*>(this->cell->clone());

}

}

DefaultAppearance.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Appearance.h"

namespace ATable {

class DefaultAppearance : public Appearance {

public:

DefaultAppearance() {

this->left\_top\_corner = 218;

this->right\_top\_corner = 191;

this->left\_bottom\_corner = 192;

this->right\_bottom\_corner = 217;

this->top\_separator = 194;

this->bottom\_separator = 193;

this->left\_separator = 195;

this->right\_separator = 180;

this->center\_separator = 197;

this->vertical\_line = 179;

this->horizontal\_line = 196;

}

};

}

DefaultCells.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Cell.h"

#include <cmath>

namespace ATable {

class StringCell : public Cell {

public:

string value;

public:

StringCell(string value) {

this->value = value;

}

StringCell(StringCell\* input) {

this->value = input->value;

}

string output() {

return this->value;

}

virtual TableObject\* clone() {

Cell\* clone = new StringCell(this->value);

clone->setWidth(this->getWidth());

return clone;

}

};

class IntegerCell : public Cell {

public:

int value;

public:

IntegerCell(int value) {

this->value = value;

}

IntegerCell(IntegerCell\* input) {

this->value = input->value;

}

string output() {

return to\_string(this->value);

}

virtual TableObject\* clone() {

Cell\* clone = new IntegerCell(this->value);

clone->setWidth(this->getWidth());

return clone;

}

};

class LongCell : public Cell {

public:

int value;

public:

LongCell(int value) {

this->value = value;

}

LongCell(LongCell\* input) {

this->value = input->value;

}

string output() {

return to\_string(this->value);

}

virtual TableObject\* clone() {

Cell\* clone = new LongCell(this->value);

clone->setWidth(this->getWidth());

return clone;

}

};

class FloatCell : public Cell {

public:

float value;

int percision;

public:

FloatCell(float value, int percision) {

this->value = value;

this->percision = percision;

}

FloatCell(float value) : FloatCell(value, 3) {}

FloatCell(FloatCell\* input) {

this->value = input->value;

this->percision = input->percision;

}

string output() {

string integer = to\_string((int)value);

string fractional = to\_string((int)(abs(value - (int)value) \* pow(10, percision)));

string result;

result += integer + "." + fractional;

return result;

}

virtual TableObject\* clone() {

Cell\* clone = new FloatCell(this->value, this->percision);

clone->setWidth(this->getWidth());

return clone;

}

};

class DoubleCell : public Cell {

public:

float value;

int percision;

public:

DoubleCell(double value, int percision) {

this->value = value;

this->percision = percision;

}

DoubleCell(double value) : DoubleCell(value, 3) {}

DoubleCell(DoubleCell\* input) {

this->value = input->value;

this->percision = input->percision;

}

string output() {

string integer = to\_string((int)value);

string fractional = to\_string((int)(abs(value - (int)value) \* pow(10, percision)));

string result;

result += integer + "." + fractional;

return result;

}

virtual TableObject\* clone() {

Cell\* clone = new DoubleCell(this->value, this->percision);

clone->setWidth(this->getWidth());

return clone;

}

};

}

IdColumn.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "DefaultCells.h"

#include "NamedColumn.h"

namespace ATable {

class IdColumn : public NamedColumn {

public:

IdColumn(unsigned int width, string name, string key);

IdColumn(unsigned int width, string key) : IdColumn(width, "ID", key) {}

IdColumn(string key) : IdColumn(3, "ID", key) {}

~IdColumn();

Cell\* getCellClone(int id); // returns new cell with id value inside

private:

Cell\* getCell(int id); // returns nullptr (not used)

};

}

IdColumn.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "IdColumn.h"

namespace ATable{

IdColumn::IdColumn(unsigned int width, string name, string key) : NamedColumn(width, name, key) {}

IdColumn::~IdColumn() {}

Cell \* IdColumn::getCell(int id) {

return nullptr;

}

Cell \* IdColumn::getCellClone(int id) {

Cell\* temp = new IntegerCell(id);

temp->setWidth(this->getWidth());

return temp;

}

}

Keyable.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include <string>

namespace ATable {

using namespace std;

class Keyable {

protected:

string key;

public:

Keyable(string key);

~Keyable();

string getKey(); // returns key key

};

}

Keyable.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Keyable.h"

namespace ATable {

Keyable::Keyable(string key) {

this->key = key;

}

Keyable::~Keyable() {}

string Keyable::getKey() {

return this->key;

}

}

Nameable.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include <string>

namespace ATable {

using namespace std;

class Namable {

public:

string name;

public:

Namable(string name);

~Namable();

string getName(); // returns name name

void setName(string name); // sets name

};

}

Nameable.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Namable.h"

namespace ATable {

Namable::Namable(string name) {

this->name = name;

}

Namable::~Namable() {}

string Namable::getName() {

return this->name;

}

void Namable::setName(string name) {

this->name = name;

}

}

NamedColumn.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Column.h"

#include "Namable.h"

namespace ATable {

class NamedColumn : public Column, public Namable {

public:

NamedColumn(unsigned int width, string name, string key);

~NamedColumn();

};

}

NamedColumn.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "NamedColumn.h"

namespace ATable {

NamedColumn::NamedColumn(unsigned int width, string name, string key) : Column(width, key), Namable(name) {}

NamedColumn::~NamedColumn() {}

}

SimpleColumn.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Cell.h"

#include "NamedColumn.h"

#include <vector>

namespace ATable {

class SimpleColumn : public NamedColumn{

public:

string name;

protected:

vector<Cell\*>\* cells;

public:

SimpleColumn(string name, unsigned int width, string key);

~SimpleColumn();

void addCell(Cell\* cell); // adds cell cell to column

void editCell(size\_t row\_id, Cell\* cell); // edits cell cell to column

Cell\* getCell(int id); // returns cell on position id

Cell\* getCellClone(int id); // returns new instance of cell on position id

unsigned int getHeight(); // returns height of column (number of cells inside)

};

}

SimpleColumn.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "SimpleColumn.h"

namespace ATable {

SimpleColumn::SimpleColumn(string name, unsigned int widht, string key) : NamedColumn(width, name, key){

this->width = widht;

this->cells = new vector<Cell\*>();

}

SimpleColumn::~SimpleColumn() {

for (unsigned int i = 0; i < this->cells->size(); i++) {

delete this->getCell(i);

}

delete this->cells;

}

void SimpleColumn::addCell(Cell\* cell) {

cell->setWidth(this->width);

this->cells->push\_back(cell);

}

void SimpleColumn::editCell(size\_t row\_id, Cell\* cell) {

cell->setWidth(this->width);

try {

delete this->cells->at(row\_id);

this->cells->at(row\_id) = cell;

}

catch (out\_of\_range e) {

throw WrongIDException();

}

}

Cell\* SimpleColumn::getCell(int id) {

if (id >= this->cells->size()) {

throw WrongIDException();

}

Cell\* cell = (\*(this->cells))[id];

return cell;

}

Cell \* SimpleColumn::getCellClone(int id) {

try {

Cell\* cell = this->getCell(id);

Cell\* temp = dynamic\_cast<Cell\*>(cell->clone());

return temp;

}

catch (WrongIDException e) {

throw e;

}

}

unsigned int SimpleColumn::getHeight() {

return this->cells->size();

}

}

Table.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "TableObject.h"

#include "SimpleColumn.h"

#include "TableExceptions.h"

#include "Appearance.h"

#include <vector>

#include <iostream>

#include <cmath>

namespace ATable {

class Table : public TableObject {

public:

vector<Column\*>\* columns;

Appearance preset;

string name;

public:

Table(Appearance preset, string name);

~Table();

void addColumn(Column\* column); // adds column column to table

void addCell(int column\_id, Cell\* cell); // adds cell cell to column with id column

void addCell(string column\_key, Cell\* cell); // adds cell cell to column with key column\_key

void editCell(string column\_key, size\_t cell\_id, Cell\* cell); // edit cell cell in column with key column\_key and row position cell\_id

void editCell(size\_t column\_id, size\_t cell\_id, Cell\* cell); // edit cell cell in column with key column\_key and row position cell\_id

virtual void print(ostream& stream); // prints table to stream stream

Column\* getColumn(int id); // returns column on position id

Column\* getColumn(string key); // returns column with key key

private:

int getColumnId(string key); // returns id of column with key key

void drawLine(ostream & stream, char left\_corner, char horizontal\_line, char center\_separator, char right\_corner); // draws line to stream

void drawRow(ostream & stream, unsigned int row, char vertical\_line); // draws row of values to stream

void drawColumnsHeaders(ostream& stream, char vertical\_line); // draws columns headers to stream

unsigned int getMaxRowHeight(); // returns height of the highest column in table

public:

static string boundary(string value, int width); // crops value by width width and returns cropped string

public:

friend ostream& operator<<(ostream& stream, Table& table);

};

}

Table.cpp

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include "Table.h"

namespace ATable {

Table::Table(Appearance preset, string name) {

this->preset = preset;

this->name = name;

this->columns = new vector<Column\*>();

}

Table::~Table() {

for (unsigned int col = 0; col < this->columns->size(); col++) {

Column\* temp = this->getColumn(col);

delete temp;

}

delete this->columns;

}

void Table::addColumn(Column\* column) {

this->columns->push\_back(column);

}

void Table::addCell(int column, Cell\* cell) {

try {

SimpleColumn\* scolumn = dynamic\_cast<SimpleColumn\*>(this->getColumn(column));

if (scolumn) {

scolumn->addCell(cell);

}

}

catch (WrongIDException e){

}

}

void Table::addCell(string column\_key, Cell\* cell) {

int id = this->getColumnId(column\_key);

if (id == -1) {

throw KeyNotExistException();

}

this->addCell(id, cell);

}

void Table::editCell(string column\_key, size\_t cell\_id, Cell\* cell) {

int id = this->getColumnId(column\_key);

if (id == -1) {

throw KeyNotExistException();

}

this->editCell(id, cell\_id, cell);

}

void Table::editCell(size\_t column\_id, size\_t cell\_id, Cell\* cell) {

try {

SimpleColumn\* scolumn = dynamic\_cast<SimpleColumn\*>(this->getColumn(column\_id));

if (scolumn) {

scolumn->editCell(cell\_id, cell);

}

}

catch (WrongIDException e) {

}

}

void Table::print(ostream & stream) {

try {

//Top line

this->drawLine(stream, this->preset.left\_top\_corner, this->preset.horizontal\_line, this->preset.top\_separator, this->preset.right\_top\_corner);

//Output headers

this->drawColumnsHeaders(stream, this->preset.vertical\_line);

this->drawLine(stream, this->preset.left\_separator, this->preset.horizontal\_line, this->preset.center\_separator, this->preset.right\_separator);

//Rows

unsigned int maxRowHeight = this->getMaxRowHeight();

for (size\_t row = 0; row < maxRowHeight; row++) {

this->drawRow(stream, row, this->preset.vertical\_line);

if (row < maxRowHeight - 1) {

this->drawLine(stream, this->preset.left\_separator, this->preset.horizontal\_line, this->preset.center\_separator, this->preset.right\_separator);

}

}

//Bottom line

this->drawLine(stream, this->preset.left\_bottom\_corner, this->preset.horizontal\_line, this->preset.bottom\_separator, this->preset.right\_bottom\_corner);

}

catch (exception e) {

stream << "Unable to print table [" << this->name << "]" << endl;

}

}

Column\* Table::getColumn(int id) {

if (!(id < this->columns->size())) {

throw WrongIDException();

}

return (\*(this->columns))[id];

}

Column\* Table::getColumn(string key) {

int id = this->getColumnId(key);

if (id == -1) {

return nullptr;

}

return this->getColumn(id);

}

int Table::getColumnId(string key) {

for (size\_t i = 0; i < this->columns->size(); i++) {

if (this->getColumn(i)->getKey() == key) {

return i;

}

}

return -1;

}

void Table::drawLine(ostream& stream, char left\_corner, char horizontal\_line, char center\_separator, char right\_corner) {

stream << left\_corner;

for (unsigned int col = 0; col < this->columns->size(); col++) {

stream << string(this->getColumn(col)->getWidth(), horizontal\_line);

if (col < this->columns->size() - 1) {

stream << center\_separator;

}

}

stream << right\_corner << endl;

}

void Table::drawRow(ostream& stream, unsigned int row, char vertical\_line) {

stream << vertical\_line;

for (unsigned int col = 0; col < this->columns->size(); col++) {

try {

Cell\* cell = this->getColumn(col)->getCellClone(row);

cell->print(stream);

delete cell;

}

catch (WrongIDException e) {

stream << string(this->getColumn(col)->getWidth(), ' ');

}

stream << vertical\_line;

}

stream << endl;

}

void Table::drawColumnsHeaders(ostream& stream, char vertical\_line) {

stream << vertical\_line;

for (unsigned int col = 0; col < this->columns->size(); col++) {

Column\* column = this->getColumn(col);

NamedColumn\* scolumn = dynamic\_cast<NamedColumn\*>(column);

if (scolumn) {

stream << boundary(scolumn->getName(), scolumn->getWidth());

}

else {

stream << string(column->getWidth(),' ');

}

stream << vertical\_line;

}

stream << endl;

}

unsigned int Table::getMaxRowHeight() {

unsigned int maxHeight = 0;

for (size\_t i = 0; i < this->columns->size(); i++) {

SimpleColumn\* scolumn = dynamic\_cast<SimpleColumn\*>(this->getColumn(i));

if (scolumn) {

unsigned int temp = scolumn->getHeight();

if (maxHeight < temp) {

maxHeight = temp;

}

}

}

return maxHeight;

}

string Table::boundary(string value, int width) {

if (value.length() < width) {

return value + string(width - value.length(), ' ');

}

if (value.length() > width) {

string temp = value;

temp.resize(width);

return temp;

}

return value;

}

ostream& operator<<(ostream& stream, Table& table) {

table.print(stream);

return stream;

}

}

TableExceptions.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include <exception>

namespace ATable {

class ATableException : public exception {

public:

const char\* what() const noexcept override { return "Error ATableException"; }

};

class WrongIDException : public ATableException {

public:

const char\* what() const noexcept override { return "Error WrongIDException"; }

};

class KeyNotExistException : public ATableException {

public:

const char\* what() const noexcept override { return "Error KeyNotExistException"; }

};

class WrongColumnTypeException : public ATableException {

public:

const char\* what() const noexcept override { return "Error WrongColumnTypeException"; }

};

}

TableObject.h

#pragma once

/\*

Appearance.cpp, Cell.cpp Column.cpp, ConstColumn.cpp IdColumn.cpp Keyable.cpp

Nameable.cpp NamedColumn.cpp SimpleColumn.cpp Table.cpp TableObject.cpp

Appearance.h ATble.h Cell.h Column.h ConstColumn.h DefaultAppearance.h DefaultCells.h

IdColumn.h Keyable.h Nameable.h NamedColumn.h SimpleColumn.h Table.h TableExceptions.h TableObject.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.0.2

\*/

#include <windows.h>

namespace ATable {

using namespace std;

class TableObject {

public:

TableObject();

virtual ~TableObject();

virtual TableObject\* clone();

};

}

WonderMenu library

MenuCommand.h

#pragma once

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include <string>

#include "WonderMenuExceptions.h"

namespace WMenu {

using namespace std;

class MenuCommand {

protected:

wstring key;

wstring description;

public:

MenuCommand(wstring key, wstring description);

virtual ~MenuCommand();

wstring getKey();

wstring getDescription();

public:

virtual void handleCommnad(wstring inputData) = 0;

};

}

MenuComamnd.cpp

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include "MenuCommand.h"

namespace WMenu {

MenuCommand::MenuCommand(wstring key, wstring description) : key(key), description(description) {}

MenuCommand::~MenuCommand() {}

wstring MenuCommand::getKey() {

return this->key;

}

wstring MenuCommand::getDescription() {

return this->description;

}

}

MenuCommandEmpty.h

#pragma once

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include "MenuCommand.h"

namespace WMenu {

class MenuCommandEmpty : public MenuCommand {

public:

MenuCommandEmpty(wstring key) : MenuCommand(key, L"") {}

~MenuCommandEmpty() {}

public:

void handleCommnad(wstring inputData) override {}

};

}

WMenu.h

#pragma once

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include "WonderMenu.h"

#include "WonderMenuExceptions.h"

#include "MenuCommand.h"

WonderMenu.h

#pragma once

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include <set>

#include <string>

#include <iostream>

#include "MenuCommand.h"

#include "WonderMenuExceptions.h"

namespace WMenu {

using namespace std;

typedef set<MenuCommand\*, bool(\*)(MenuCommand\* less, MenuCommand\* higher)> MSet;

class WonderMenu : private MSet {

protected:

wstring exitCommand;

public:

WonderMenu(wstring exitCommand);

WonderMenu();

virtual ~WonderMenu();

void addCommand(MenuCommand\* command);

void start();

public:

virtual void print() = 0;

public:

using MSet::size;

using MSet::begin;

using MSet::end;

using MSet::iterator;

private:

MenuCommand\* find(wstring key);

static bool comparator(MenuCommand\* less, MenuCommand\* higher);

};

}

WonderMenu.h

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include "WonderMenu.h"

#include "MenuCommandEmpty.h"

namespace WMenu {

WonderMenu::WonderMenu(wstring exitCommand) : MSet(&comparator), exitCommand(exitCommand) {}

WonderMenu::WonderMenu() : WonderMenu(L"exit") {}

WonderMenu::~WonderMenu() {

for (MenuCommand\* item : \*this) {

delete item;

}

}

void WonderMenu::addCommand(MenuCommand\* command) {

if (!command) {

throw ZeroPointerException();

}

try {

delete this->find(command->getKey());

this->insert(command);

}

catch (NoObjectFoundException e) {

this->insert(command);

}

}

void WonderMenu::start() {

while (true) {

this->print();

wstring buffer;

wstring command;

wcin >> command;

for (size\_t i = command.length(); i > 0 ; i--) {

wcin.putback(command[i-1]);

}

getline(wcin, buffer);

try {

MenuCommand\* item = this->find(command);

item->handleCommnad(buffer);

}

catch (NoObjectFoundException e) {

wcout << L"Error: Unknown command" << endl;

}

if (command.compare(this->exitCommand) == 0) {

break;

}

}

}

MenuCommand\* WonderMenu::find(wstring key) {

MenuCommandEmpty finder(key);

iterator item = MSet::find(&finder);

if (item == this->cend()) {

throw NoObjectFoundException();

}

return \*item;

}

bool WonderMenu::comparator(MenuCommand\* less, MenuCommand\* higher) {

int comparingResult = less->getKey().compare(higher->getKey());

if (comparingResult == -1) {

return true;

}

return false;

}

}

WonderMenuExceptions.h

#pragma once

/\*

MenuCommand.cpp, WonderMenu.cpp MenuCommand.h MenuCommandEmpty.h WonderMenu.h WonderMenuExceptions.h WMenu.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.0

\*/

#include <exception>

namespace WMenu {

class WonderMenuException : public std::exception {

public:

const char\* what() const noexcept override { return "Error: WonderMenuException"; }

};

class NoObjectFoundException : public WonderMenuException {

public:

const char\* what() const noexcept override { return "Error: NoObjectFoundException"; }

};

class ZeroPointerException : public WonderMenuException {

public:

const char\* what() const noexcept override { return "Error: ZeroPointerException"; }

};

}

TurboPipes library

Pipeable.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include <queue>

#include <string>

#include "Threadable.h"

namespace TurboPipes {

using namespace std;

class PipeDispatcher;

class Pipeable : private Threadable {

private:

queue<byte\*> messages;

protected:

PipeDispatcher\* dispatcher;

public:

Pipeable();

~Pipeable();

void connect(PipeDispatcher\* dispatcher);

void queueMessage(byte\* message);

using Threadable::getThreadHandle;

private:

void threadFunction() override;

void handler();

public:

virtual void handleMessage(byte\* message) = 0;

};

}

Pipeable.cpp

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "Pipeable.h"

namespace TurboPipes {

Pipeable::Pipeable() : Threadable() {

this->dispatcher = nullptr;

}

Pipeable::~Pipeable() {}

void Pipeable::connect(PipeDispatcher\* dispatcher) {

this->dispatcher = dispatcher;

}

void Pipeable::queueMessage(byte\* message) {

this->messages.push(message);

}

void Pipeable::threadFunction() {

this->handler();

}

void Pipeable::handler() {

while (this->isRunning()) {

if (this->messages.empty() || !this->dispatcher) {

Sleep(10);

}

else {

byte\* message = this->messages.back();

this->messages.pop();

this->handleMessage(message);

}

}

}

}

PipeableString.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "Pipeable.h"

namespace TurboPipes {

class PipeDispatcherString;

class PipeableString : private Pipeable {

protected:

PipeDispatcherString\* dispatcher;

public:

PipeableString();

~PipeableString();

void connect(PipeDispatcherString\* dispatcher);

void queueMessage(wstring message);

using Pipeable::queueMessage;

using Pipeable::getThreadHandle;

public:

virtual void handleMessage(wstring& message) = 0;

private:

void handleMessage(byte\* message) override;

};

}

PipeableString.cpp

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "PipeableString.h"

namespace TurboPipes {

PipeableString::PipeableString() {

this->dispatcher = nullptr;

}

PipeableString::~PipeableString() {

this->dispatcher = nullptr;

}

void PipeableString::connect(PipeDispatcherString\* dispatcher) {

this->dispatcher = dispatcher;

}

void PipeableString::queueMessage(wstring message) {

byte\* byte\_message = new byte[(message.length() + 1) \* sizeof(wchar\_t)];

Pipeable::queueMessage(byte\_message);

}

void PipeableString::handleMessage(byte\* message) {

wstring string\_message = reinterpret\_cast<wchar\_t\*>(message);

this->handleMessage(string\_message);

}

}

PipeDispathcer.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include <windows.h>

#include <string>

#include "Pipeable.h"

#include "Threadable.h"

#include "PipesExceptions.h"

#include <codecvt>

#include <iostream>

namespace TurboPipes {

using namespace std;

class PipeDispatcher : public Threadable {

private:

static const DWORD TIMEOUT = 5000;

static const DWORD BUFFER\_SIZE = 1024;

private:

HANDLE hNamedPipe;

DWORD cbWritten;

DWORD cbRead;

wstring szBuf;

wstring pipeName;

bool is\_server;

protected:

Pipeable\* object;

public:

PipeDispatcher(wstring szPipeName, bool is\_server, Pipeable\* object);

~PipeDispatcher();

void throwMessage(byte\* message, DWORD length);

protected:

byte\* catchMessage();

void messagesHandler();

void threadFunction() override;

protected: // copying restricted

PipeDispatcher(const PipeDispatcher& reference) {}

PipeDispatcher& operator= (const PipeDispatcher& reference) {}

};

}

PipeDispatcher.cpp

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "PipeDispatcher.h"

const int TIMEOUT\_CONNECT\_ATTEMPTS = 30;

const size\_t CONNECTING\_SLEEP\_TIME = 100;

namespace TurboPipes {

PipeDispatcher::PipeDispatcher(wstring szPipeName, bool is\_server, Pipeable\* object) : pipeName(szPipeName), object(object) {

this->object->connect(this);

this->cbWritten = 0;

this->cbRead = 0;

this->hNamedPipe = INVALID\_HANDLE\_VALUE;

this->is\_server = is\_server;

if (this->is\_server) {

this->hNamedPipe = CreateNamedPipe(

this->pipeName.c\_str(),

PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES,

BUFFER\_SIZE,

BUFFER\_SIZE,

TIMEOUT,

NULL);

if (this->hNamedPipe == INVALID\_HANDLE\_VALUE) {

throw PipeCreationException();

}

BOOL fConnected = ConnectNamedPipe(hNamedPipe, NULL);

if (!fConnected) {

switch (GetLastError()) {

case ERROR\_NO\_DATA:

CloseHandle(hNamedPipe);

throw PipeNoDataException();

break;

case ERROR\_PIPE\_CONNECTED:

CloseHandle(hNamedPipe);

throw PipeConnectedException();

break;

case ERROR\_PIPE\_LISTENING:

CloseHandle(hNamedPipe);

throw PipeListeningException();

break;

case ERROR\_CALL\_NOT\_IMPLEMENTED:

CloseHandle(hNamedPipe);

throw PipeCallNotImplementedException();

break;

default:

CloseHandle(hNamedPipe);

throw PipeException();

break;

}

CloseHandle(hNamedPipe);

}

}

else {

int attempts = 0;

bool connecting\_flag = true;

while (attempts <= TIMEOUT\_CONNECT\_ATTEMPTS + 1 && connecting\_flag) {

this->hNamedPipe = CreateFile(

this->pipeName.c\_str(),

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

OPEN\_EXISTING,

0,

NULL);

if (this->hNamedPipe == INVALID\_HANDLE\_VALUE && attempts >= TIMEOUT\_CONNECT\_ATTEMPTS) {

throw PipeConnectionException();

}

if (this->hNamedPipe != INVALID\_HANDLE\_VALUE) {

connecting\_flag = false;

}

}

}

}

PipeDispatcher::~PipeDispatcher() {

this->object->connect(nullptr);

CloseHandle(this->hNamedPipe);

}

void PipeDispatcher::throwMessage(byte\* message, DWORD length) {

BOOL isWritten = WriteFile(hNamedPipe, message, length, &cbWritten, NULL);

if (!isWritten) {

throw PipeWritingException();

}

}

byte\* PipeDispatcher::catchMessage() {

DWORD bytesAvail = 0;

DWORD bytesMessage = 0;

byte\* message = nullptr;

while (!bytesAvail) {

PeekNamedPipe(hNamedPipe, NULL, 0, NULL, &bytesAvail, &bytesMessage);

if (bytesAvail) {

message = new byte[bytesAvail];

BOOL isRead = ReadFile(hNamedPipe, message, bytesAvail, &cbRead, NULL);

if (!isRead) {

throw PipeReadingException();

}

return message;

}

else {

Sleep(10);

}

}

return message;

}

void PipeDispatcher::messagesHandler() {

while (this->isRunning()) {

try {

byte\* message = this->catchMessage();

this->object->queueMessage(message);

}

catch (PipeReadingException e) {

throw PipeLostConnectionException();

}

}

}

void PipeDispatcher::threadFunction() {

this->messagesHandler();

}

}

PipeDispatcherString.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "PipeDispatcher.h"

#include "PipeableString.h"

namespace TurboPipes {

class PipeDispatcherString : private PipeDispatcher {

public:

PipeDispatcherString(wstring szPipeName, bool is\_server, PipeableString\* object);

~PipeDispatcherString();

void throwMessage(wstring& message);

using PipeDispatcher::getThreadHandle;

using PipeDispatcher::stopThread;

protected:

wstring catchMessage();

protected:

PipeDispatcherString(const PipeDispatcherString& reference) : PipeDispatcher(reference) {}

PipeDispatcherString& operator= (const PipeDispatcherString& reference) {}

};

}

PipeDispatcherString.cpp

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "PipeDispatcherString.h"

namespace TurboPipes {

PipeDispatcherString::PipeDispatcherString(wstring szPipeName, bool is\_server, PipeableString\* object) : PipeDispatcher(szPipeName, is\_server, (Pipeable\*)object) {

object->connect(this);

}

PipeDispatcherString::~PipeDispatcherString() {}

void PipeDispatcherString::throwMessage(wstring& message) {

PipeDispatcher::throwMessage((byte\*)message.c\_str(), (message.length() + 1) \* sizeof(wchar\_t));

}

wstring PipeDispatcherString::catchMessage() {

byte\* byte\_massage = PipeDispatcher::catchMessage();

wstring string\_message((wchar\_t\*)byte\_massage);

delete[] byte\_massage;

return string\_message;

}

}

PipesExceptions.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include <exception>

namespace TurboPipes {

using namespace std;

class PipeException : public exception {

const char\* what() const noexcept override { return "Error: PipeException"; }

};

class PipeCreationException : public PipeException {

const char\* what() const noexcept override { return "Error: PipeCreationException"; }

};

class PipeConnectionException : public PipeException {

const char\* what() const noexcept override { return "Error: PipeConnectionException"; }

};

class PipeWritingException : public PipeException {

const char\* what() const noexcept override { return "Error: PipeWritingException"; }

};

class PipeNoDataException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeNoDataException"; }

};

class PipeConnectedException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeConnectedException"; }

};

class PipeListeningException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeListeningException"; }

};

class PipeCallNotImplementedException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeCallNotImplementedException"; }

};

class PipeReadingException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeReadingException"; }

};

class PipeLostConnectionException : public PipeConnectionException {

const char\* what() const noexcept override { return "Error: PipeLostConnectionException"; }

};

}

Threadable.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include <windows.h>

#include <process.h>

namespace TurboPipes {

class Threadable {

private:

bool running;

HANDLE threadHandle;

public:

Threadable();

virtual ~Threadable();

void startThread();

void stopThread();

HANDLE getThreadHandle();

bool isRunning();

virtual void threadFunction() = 0;

private:

static unsigned int \_stdcall recieveMessageThread(void\* pthis);

void threadFunctionDispathcer();

};

}

Threadable.cpp

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "Threadable.h"

namespace TurboPipes {

Threadable::Threadable() {

this->running = false;

this->threadHandle = INVALID\_HANDLE\_VALUE;

this->startThread();

}

Threadable::~Threadable() {

this->running = false;

}

void Threadable::startThread() {

this->running = true;

this->threadHandle = (HANDLE)\_beginthreadex(0, 0, &Threadable::recieveMessageThread, this, 0, 0);

}

void Threadable::stopThread() {

this->running = 0;

}

HANDLE Threadable::getThreadHandle() {

return this->threadHandle;

}

bool Threadable::isRunning() {

return this->running;

}

unsigned int \_stdcall Threadable::recieveMessageThread(void\* pthis) {

Threadable\* pThreadable = static\_cast<Threadable\*>(pthis);

pThreadable->threadFunction();

delete pThreadable;

return 0;

}

void Threadable::threadFunctionDispathcer() {

this->threadFunction();

\_endthreadex(0);

CloseHandle(threadHandle);

}

}

TurboPipes.h

#pragma once

/\*

Pipeable.cpp, PipeableString.cpp PipeDispatcher.cpp, PipeDispatcherString.cpp Threadable.cpp

Pipeable.h PipeableString.h PipeDispatcher.h PipeDispatcherString.h PipesException.h Threadable.h TurboPipes.h

Copyright (C) 2019 Andrieiev Danil

This source code is provided 'as-is', without any express or implied

warranty. In no event will the author be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not

claim that you wrote the original source code. If you use this source code

in a product, an acknowledgment in the product documentation would be

appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be

misrepresented as being the original source code.

3. This notice may not be removed or altered from any source distribution.

Andrieiev Danil danssg08@gmail.com

version 1.1.3

\*/

#include "PipesExceptions.h"

#include "PipeDispatcherString.h"

#include "PipeableString.h"

#include "PipeDispatcher.h"

#include "Pipeable.h"