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;

}

}

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);

public:

using vector<Fileable\*>::operator[];

};