

Московский государственный университет имени М.В. Ломоносова

Факультет Вычислительной математики и кибернетики

Кафедра Алгоритмических языков

Отчет по заданию практикума по объектно-ориентированному программированию

**Система управления инвестиционным портфелем**

**Выполнил:**

студент 424 группы

**Гончаренко Евгений**

Москва, 2023

Оглавление

[Уточнение постановки задачи 3](#__RefHeading___Toc81495_2230964193)

[Диаграмма классов 3](#__RefHeading___Toc81497_2230964193)

[Текстовые спецификации интерфейса 4](#__RefHeading___Toc81499_2230964193)

[Инструментальные средства 10](#__RefHeading___Toc81501_2230964193)

[Описание файловой структуры программы 10](#__RefHeading___Toc81503_2230964193)

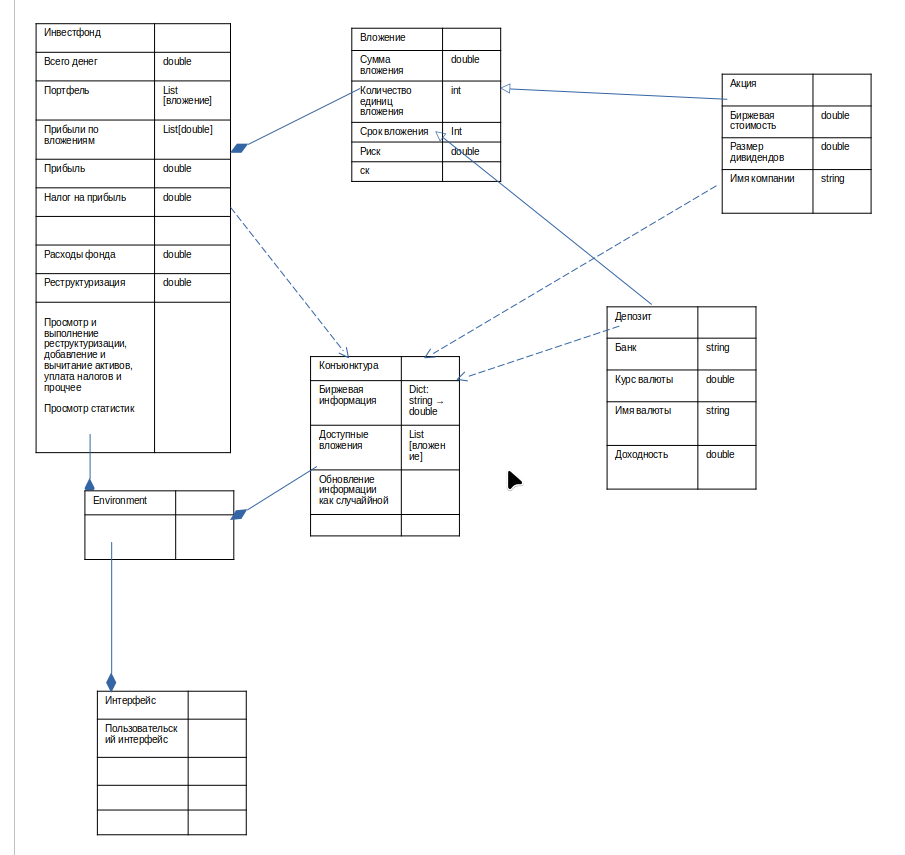
[Пользовательский интерфейс 11](#__RefHeading___Toc81505_2230964193)

# **Уточнение постановки задачи**

Задача – создать Систему управления инвестиционным портфелем, позволяющую:

* Вводить цены на активы и содержимое портфеля в json-файле, в том числе, цену актива, дивиденты и риски активов
* Отслеживать изменения цен на активы
* Эмулировать по крайней мере два тренды на рынке: бычий и медвежий
* Обеспечивающую дружественный пользователю вывод информации о состояния портфеля, извлеченных дивидендах и общей прибыли

# **Диаграмма классов**



# **Текстовые спецификации интерфейса**

Класс конъюнктура

|  |
| --- |
| class Conjuncture : public std::enable\_shared\_from\_this<Conjuncture>  {  *Цены риски и цены активов, тренд(бычий или медвежий)*  std::map<std::string, double> market\_cost;  std::map<std::string, double> market\_risks;  char trend = 'B';  public:  *Конструктор*  Conjuncture(json config, char trend\_passed = 'B');  *Совершить шаг моделирования*  void make\_time\_step();  *Получить рыночную цену актива*  double get\_market\_cost(std::string asset\_name);  ~Conjuncture()  }; |

Класс вложение

|  |
| --- |
| class Investition  {  *Цена и риск единицы вложения; количество единиц вложения*  protected:  double cost = 0; //calculated due to market info  double risk = 0; //attached to an asset  int quantity = 0;  *Связь с конъюнктурой*  std::shared\_ptr<Conjuncture> conjuncture\_ptr = nullptr;  public:  *Конструктор*  Investition( json & invest\_json, std::shared\_ptr<Conjuncture> conjuncture\_ptr\_passed)  double get\_cost()  double get\_risk()  double get\_investition\_time()  void set\_investition\_time(double var)  void set\_cost(double var)  void set\_risk(double var)  ~Investition()  *Получить обновление цены от конъюнктуры*  virtual void get\_update\_from\_conjuncture() = 0;  *Получить имя актива*  virtual std::string get\_name() = 0;  *Извлечь прибыль от актива*  virtual double extract\_profit() = 0;  }; |

Класс вклад

|  |
| --- |
| class Deposit : public Investition {  *Имя банка и название валюты хранения, процент по вкладу*  private:  std::string bank\_name;  double currency\_value = 0;  std::string currency\_name;  double deposit\_percent = 0;  public:  Deposit( json invest\_json, std::shared\_ptr<Conjuncture> conjuncture\_ptr\_passed ) :  std::string get\_name()  std::string get\_bank\_name()  double get\_deposit\_percent()  void set\_deposit\_percent(double var)  double extract\_profit()  json close\_deposit()  ~Deposit()  void get\_update\_from\_conjuncture()  }; |

Класс акция

|  |
| --- |
| class Stock : public Investition {  private:  *Имя компани и процент по дивиденду*  std::string company\_name;  //One market share cost  double share\_market\_cost = 0;  double divident\_percent = 0;  public:  *Конструктор*  Stock( json invest\_json, std::shared\_ptr<Conjuncture> conjuncture\_ptr\_passed ) :  std::string get\_name()  double get\_share\_market\_cost()  double set\_share\_market\_cost(double var)  double get\_divident\_percent()  double set\_divident\_percent(double var)  double extract\_profit()  void get\_update\_from\_conjuncture()  }; |

Класс инвестфонд

|  |
| --- |
| class InvestFund  {  private:  *Список вложений*  std::vector<std::shared\_ptr<Investition>> fund\_briefcase;  *Накопленные извлесенные прибыли*  std::vector<double> briefcase\_total\_profits;  *Изначальные стоимости*  std::vector<double> initial\_costs;  *Связь с конъюнктурой*  std::shared\_ptr<Conjuncture> conjuncture\_ptr = nullptr;  public:  *Конструктор*  InvestFund( nlohmann::json invest\_fund\_config, std::shared\_ptr<Conjuncture> conjuncture\_ptr\_passed)  *Сделать шаг по времени*  void make\_time\_step()  *Геттеры: имен активов, их количества, текущих стоимостей, накопленных извлеченных прибылей, изначальных стоимотей, дохода(убытка) от владения*  std::vector<std::string> get\_names()  int get\_q\_briefcase()  std::vector<double> get\_costs()  std::vector<double> get\_profits()  std::vector<double> get\_initial\_costs()  std::vector<double> get\_won\_by\_buying()  }; |

Класс окружение

|  |
| --- |
| class Environment  {  unique\_ptr<InvestFund> investfund\_ptr = nullptr;  shared\_ptr<Conjuncture> conjuncture\_ptr = nullptr;  public:  *Конструктор*  Environment(json got\_on\_input, char trend)  *Сделать шаг во времени для конъюнктуры и фонда*  void step()  *Геттеры: имен активов, их количества, текущих стоимостей, накопленных извлеченных прибылей, изначальных стоимотей, дохода(убытка) от владения, реально будут запрошены геттеры у единственного фонда, который моделируется*  std::vector<std::string> get\_names()  int get\_q\_briefcase()  std::vector<double> get\_costs()  std::vector<double> get\_profits()  std::vector<double> get\_initial\_costs()  std::vector<double> get\_won\_by\_buying()  }; |

# **И****нструментальные средства**

* Язык программирования – С++;
* Среда разработки – Geany;
* Используемая графическая библиотека – Qt.

# **Описание файловой структуры программы**

Корневой каталог:

* GONCHARENKO.pro
* main.cpp
* mainwindow.h
* mainwindow.cpp
* fund.cpp
* config.json

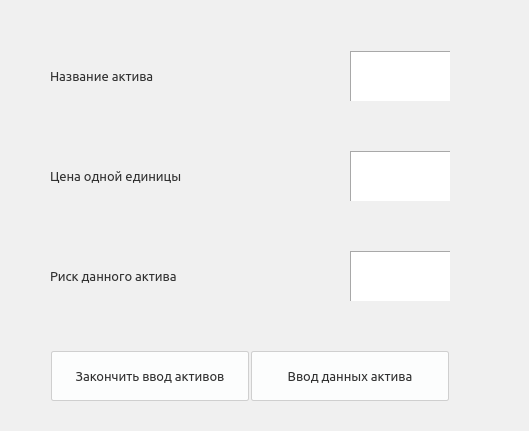
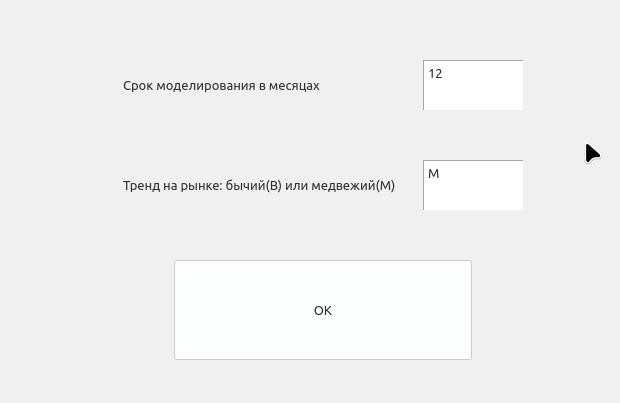
# **Пользовательский интерфейс**

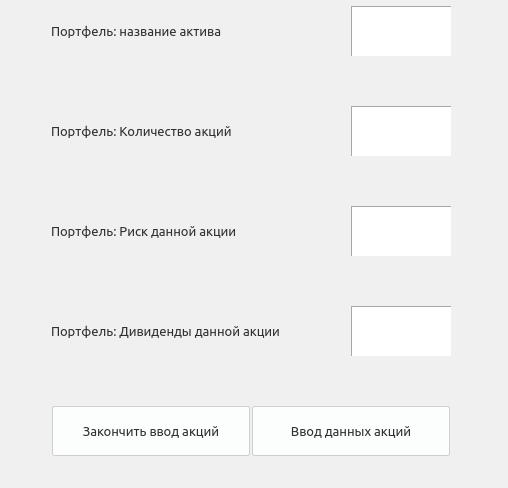
Три экрана для ввода информации:

1) о параметрах моделирования

2) о данных активов

3) о наполнении портфеля





И один экран для взаимодействия с моделированием: шаг вперед или до конца

