Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

## Курсовая работа по курсу «Программирование»

## Тема работы: «Разработка приложения с графическим интерфейсом на языке C++ с использованием библиотеки QT»

Студент: Кенть Н.В.

Преподаватель: доц. Пышкин Е.В.

Санкт-Петербург 2012

# Образ проекта.

Курсовая работа посвящена разработке компьютерного приложения «Блек-джек». В игре должны быть реализованы правила данной игры.

Правила игры «Блек-джек» :

Значения очков каждой карты: от двойки до десятки — соответственно от 2 до 10, у туза — 1 или 11 (11 пока общая сумма не больше 21, далее 1), у *картинок* (король, дама, валет) — 10.

Если у игрока и дилера число очков на руках равное, то такая ситуация называется «*Ровно*».  В такой ситуации все остаются при своих ставках, никто не выигрывает и не проигрывает. Ставки делаются до раздачи карт. После того, как первая карта сдана, игрокам запрещается делать ставки и прикасаться к своим фишкам. Если у игрока сразу после раздачи набралось 21 очко (то есть у игрока туз и 10 или туз и картинка), то такая ситуация и называется *блек-джек*. В таком случае игроку сразу выплачивается выигрыш 3 к 2. Далее игрокам, у которых не блек-джек, предлагается на выбор либо взять ещё карту (в таком случае игрок должен сказать дилеру «карту» или «ещё»), либо остаться при тех картах (и той сумме очков), которые у него на руке (в этом случае игрок должен сказать дилеру «достаточно» или «хватит»). Если у игрока после взятия новой карты сумма очков превысит 21, то такая ситуация называется «перебор». Если у дилера в первых двух картах набирается 21 очко (блек-джек), то все игроки (кроме тех, у кого тоже блек-джек), проигрывают. Те, у которых блек-джек, остаются при своих ставках. Общее правило блек-джека состоит в том, что дилер обязан остановиться, как только наберёт 17 очков или выше, и обязан брать, пока не достигнет (даже если у всех не перебравших меньше очков).

# Техническое задание

Разработать приложение на C++ игра «Блек-джек» с использованием библиотеки Qt для реализации графического интерфейса.

### Режимы работы приложения:

1. Консольное приложение
2. Интеграция кода программы для консоли с графической библиотекой Qt, и создание графического интерфейса.

### **Основные** **требования**:

В главном меню должны быть пункты:

1. Начать игру (Сесть за стол)
2. Закончить игру (Покинуть стол стола)
3. Взять еще карту
4. Закончить брать карту

### Функционирование программы:

1. Начало игры
2. Ввод имени
3. Сделать ставку
4. Раздача карт
5. Выбор : взять еще или остановиться
6. Подсчет результатов
7. Начисление выигрыша

### Требования к функциональности консольного приложения:

1. Отображать кнопки меню
2. Отображать стол и карты
3. Отображать счет

### Обработка ошибок:

1. Если игрок не ввел имя, то автоматически присваивается имя Player
2. Если игрок не ввел ставку, то поступит повторное предложение

# Проектирование системной архитектуры

### Классы приложения:

1. Класс игрок: имя игрока, функции хода, принятие решения брать еще карту, или остановится.
2. Класс дилер: аналогичные функции с учетом особенностей дилера.
3. Класс колода: реализуются функции создание колоды, тасование карт. Колода- стандартный контейнер (deque), содержащий масть и значение карты. (с использованием перечисления)
4. Класс игра: идет проверка на правильность хода, проверка на победу кого-либо, подсчет очков
5. Класс карта: создание карты (масть и значение)

### Взаимодействие классов:

Класс card устанавливает масть и номинал карты. Класс deck из полученных карт создает колоду , перетасовывает ее. Затем классы RealPlayer и Dealer (производные от класса Player) берут карты из полученной колоды, делают ставки, вводят имена. Класс Game устанавливает правила игры, по которым игроки берут карты и делают ставки, ведет подсчет очков.

Базовый класс player

Класс deck

Класс card

Класс Game

Протзводный класс Dealer

Производный класс RealPlayer

### Классы консольного приложения

В классе card создается карта, а именно, присваивается масть и значение.

1. class card

{

private:

Suits su;

Values val;

public:

void set\_suit (Suits s);

void set\_value (Values v);

QString get\_suit();

QString get\_score ();

};

В классе deck создается колода из полученных карт, тасуется.

1. class deck

{

private:

deque<card> Deck;

card crd;

void shuffle ();

public:

void create\_deck ();

card pull\_card ();

};

Класс player – создает игрока, который может брать карты, показывать их, вводить имя. Этот класс имеет два производных: RealPlayer, Dealer которые отвечают за игру реального игрока и автоматического дилера соответственно.

1. class player

{

protected:

deque <card> hand;

deck mydeck;

public:

int \*score;

int bet;

QString plname;

player ();

void enterName(QString name);

virtual int *makeBets* (int bets);

void takeCards ();

virtual void *show*(QPainter &painter);

int countScore ();

};

Класс Game отвечает за основную логику игры – определение победителя, раздача карт, подсчет очков.

1. class Game

{

private:

dialog dlg;

int \*pscore, \*dscore, cash;

QString n;

QPainter p;

public:

Game();

RealPlayer \*plr;

Dealer \*dlr;

void takeCards();

int playerScore();

int dealerScore();

void firstDispensation();

void dealerTakeCards();

int whoWon();

int setCash(int bet);

RealPlayer getPlayer();

};

#### Функционал игры

* ввод имени
* ввод ставки
* возможность брать карты из колоды
* игра подчитывает количество очков у игрока и дилера(компьютера), затем определяет победителя

### Классы графического приложения

* Начальный класс приложения, спрашивающий у пользователя о желании начать игру.

class want\_to\_play;

}

class want\_to\_play : public QDialog

{

Q\_OBJECT

public:

explicit want\_to\_play(QWidget \*parent = 0);

~*want\_to\_play*();

firstDialog \*dialog;

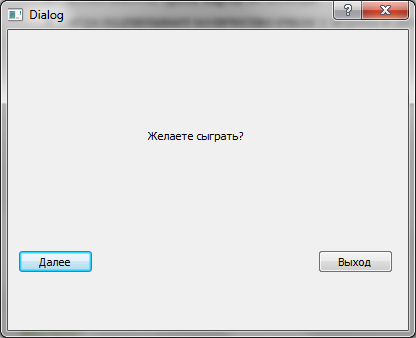
private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

private:

Ui::want\_to\_play \*ui;

};

* Окно приложения с запросом ввести имя и стаку

class firstDialog;

}

class firstDialog : public QDialog

{

Q\_OBJECT

public:

explicit firstDialog(QWidget \*parent = 0);

~*firstDialog*();

private slots:

void on\_pushButton\_3\_clicked();

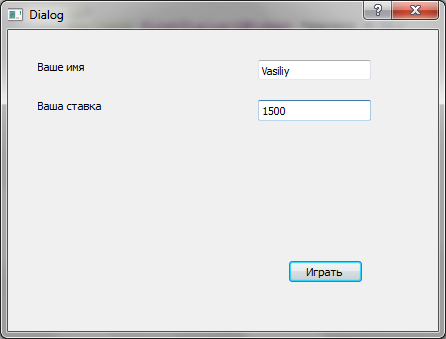
private:

Ui::firstDialog \*ui;

MainWindow \*wn;

QString name;

};



* Класс окна с картами и очками пользователя

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

void *paintEvent*(QPaintEvent \*event);

dealer\_cards \*dealer\_crd;

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

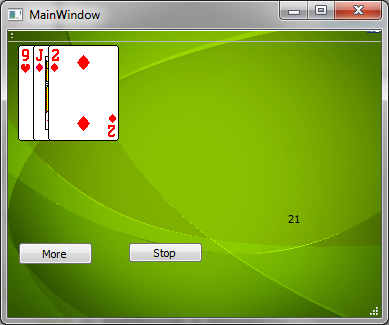
private:

Ui::MainWindow \*ui;

QImage im;

QPainter pen;

};



* Класс окна с картами и очками пользователя

class dealer\_cards;

}

class dealer\_cards : public QDialog

{

Q\_OBJECT

public:

explicit dealer\_cards(QWidget \*parent = 0);

~*dealer\_cards*();

void *paintEvent*(QPaintEvent \*ev);

Game \*gm;

information \*info;

private slots:

void on\_pushButton\_clicked();

private:

Ui::dealer\_cards \*ui;

QPainter qp;

QImage image;

};



* Класс окна с результатами

class information : public QDialog

{

Q\_OBJECT

public:

explicit information(QWidget \*parent = 0);

~*information*();

QLabel \*getLabel();

QLabel \*getLabel2();

QLabel \*getLabel3();

QLabel \*getLabel4();

private slots:

void on\_pushButton\_clicked();

private:

Ui::information \*ui;

};



* Класс окна, спрашивающего у пользователя, желает ли он сыграть еще раз

class play\_again;

}

class play\_again : public QDialog

{

Q\_OBJECT

public:

explicit play\_again(QWidget \*parent = 0);

~*play\_again*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

private:

Ui::play\_again \*ui;

};

