



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования «Московский государственный  
технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»**

**Лабораторная работа №6**

**«Разработка бота на основе конечного автомата для Telegram с  
использованием языка Python»**

**по дисциплине «Базовые компоненты интернет-технологий»**

**Выполнила:  
студентка группы ИУ5-35Б  
Ищенко А.С.**

**2021 г.**

**Цель лабораторной работы:** изучение разработки ботов в Telegram.

**Задание:**

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

## Текст программы:

### main.py

```
import os
import telebot
from telebot import types
from game import Game
import random

# Токент бота
TOKEN = '5095553179:AAHkhDObVrZH7LWnnZ9x10yMny35xc_y1g8'

# Создание бота
bot = telebot.TeleBot(TOKEN)

game = None

cur_path = os.path.dirname(os.path.abspath(__file__))

@bot.message_handler(commands=['start'])
def cmd_start(message):
    bot.send_message(message.chat.id, 'Добро пожаловать в наше заведение!')
    markup_inline = types.InlineKeyboardMarkup()
    button_yes = types.InlineKeyboardButton(text='Согласиться', callback_data="Agree")
    button_no = types.InlineKeyboardButton(text='Отказаться', callback_data="Disagree")
    markup_inline.add(button_yes, button_no)
    bot.send_message(message.chat.id, 'Сыграете в блэкджек?', reply_markup=markup_inline)

@bot.callback_query_handler(func=lambda c: c.data == 'Agree')
def handle_agree(callback_query: types.CallbackQuery):
    global game
    if game is not None:
        bot.send_message(callback_query.from_user.id, 'Игра уже начата')
        return
    game = Game()

    bot.send_message(callback_query.from_user.id, 'Ваши карты:' + "\n" + game.client.print_cards())
    bot.send_message(callback_query.from_user.id, "Очки: " + str(game.client.sum()))
    winner = check_finish()
    if winner is not None:
        congratulation_message(callback_query, winner)
        return

    bot.send_message(callback_query.from_user.id, 'Карта бота:' + "\n" + game.bot.print_cards(first_card=True))
    # bot.send_message(callback_query.from_user.id, "Bot " + str(game.bot.sum()))
    winner = check_finish()
    if winner is not None:
```

```

        congratulation_message(callback_query, winner)
        return

    else:
        markup_inline = send_question()
        bot.send_message(callback_query.from_user.id, 'Еще?',
reply_markup=markup_inline)

@bot.callback_query_handler(func=lambda c: c.data == 'Disagree')
def handle_disagree(callback_query: types.CallbackQuery):
    finish(callback_query)

@bot.callback_query_handler(func=lambda c: c.data == 'No')
def handle_no(callback_query: types.CallbackQuery):
    if game is None:
        return
    winner = check_finish()

    if winner is None:
        if game.bot.sum() > game.client.sum():
            message = 'Победитель бот :('
        elif game.bot.sum() < game.client.sum():
            message = 'Вы победитель :)'
        elif game.bot.sum() == game.client.sum():
            message = 'Ничья :|'
        bot.send_message(callback_query.from_user.id, message)
        open_the_cards(callback_query)
        finish(callback_query)
        return
    congratulation_message(callback_query, check_finish())
    return

@bot.callback_query_handler(func=lambda c: c.data == 'Yes')
def handle_yes(callback_query: types.CallbackQuery):
    if five_cards(callback_query): return

    add_cards_to_client()
    bot.send_message(callback_query.from_user.id, 'Ваши карты:' + "\n"
+ game.client.print_cards())
    bot.send_message(callback_query.from_user.id, "Очки: " +
str(game.client.sum()))
    winner = check_finish()
    if winner is not None:
        congratulation_message(callback_query, winner)
        return

    add_cards_to_bot()
    # bot.send_message(callback_query.from_user.id, "Bot " +
str(game.bot.sum()))
    winner = check_finish()
    if winner is not None:
        congratulation_message(callback_query, winner)
        return

    markup_inline = send_question()
    bot.send_message(callback_query.from_user.id, 'Еще?',
reply_markup=markup_inline)

```

```

def send_question():
    markup_inline = types.InlineKeyboardMarkup()
    button_yes = types.InlineKeyboardButton(text='Да',
callback_data="Yes")
    button_no = types.InlineKeyboardButton(text='Нет',
callback_data="No")
    markup_inline.add(button_yes, button_no)
    return markup_inline

# доделать
def check_finish():
    bot_sum = game.bot.sum()
    client_sum = game.client.sum()

    if client_sum == 21:
        return 'client'
    elif bot_sum > 21:
        return 'client'
    elif bot_sum == 21:
        return 'bot'
    elif client_sum > 21:
        return 'bot'

    return None

def add_cards_to_client():
    game.add_new_cards_to_client()

def add_cards_to_bot():
    game.add_new_cards_to_bot()

def five_cards(callback_query):
    if game.client.count_cards() >= 5:
        bot.send_message(callback_query.from_user.id, 'У Вас 5 карт!
Вскрываемся!')
        congratulation_message(callback_query, check_finish())
        return True

def congratulation_message(callback_query, winner):
    bot.send_message(callback_query.from_user.id, 'Победитель ' +
winner + ' !')
    open_the_cards(callback_query)
    finish(callback_query)
    return

def open_the_cards(callback_query):
    bot.send_message(callback_query.from_user.id,
'_____')
    bot.send_message(callback_query.from_user.id, '\nИТОГИ:\n')
    bot.send_message(callback_query.from_user.id, 'Ваши карты: ' + "\n"
+ game.client.print_cards())
    bot.send_message(callback_query.from_user.id, "Очки: " +
str(game.client.sum()))
    bot.send_message(callback_query.from_user.id, '\nКарты бота: ' +
"\n" + game.bot.print_cards())

```

```
    bot.send_message(callback_query.from_user.id, "Очки: " +
str(game.bot.sum()))
    bot.send_message(callback_query.from_user.id,
'_____')

def finish(callback_query):
    global game
    game = None
    bot.send_message(callback_query.from_user.id, 'До свидания!\nБудем
рады видеть Вас снова')

if __name__ == '__main__':
    bot.infinity_polling()
```

## create\_cards.py

```
from card import Card

clubs_suit = 'Пики'
diamonds_suit = 'Буби'
hearts_suit = 'Черви'
spades_suit = 'Крести'

suits = [clubs_suit, diamonds_suit, hearts_suit, spades_suit]

a_number = 'Туз'
second_number = '2'
third_number = '3'
fourth_number = '4'
fifth_number = '5'
sixth_number = '6'
seventh_number = '7'
eleventh_number = '8'
nineth_number = '9'
tenth_number = '10'
king_number = 'Король'
lady_number = 'Дама'
jack_number = 'Валет'

numbers = [a_number,
            second_number,
            third_number,
            fourth_number,
            fifth_number,
            sixth_number,
            seventh_number,
            eleventh_number,
            nineth_number,
            tenth_number,
            king_number,
            lady_number,
            jack_number]

weights = {
    a_number: 11,
    second_number: 2,
    third_number: 3,
    fourth_number: 4,
    fifth_number: 5,
    sixth_number: 6,
    seventh_number: 7,
    eleventh_number: 8,
    nineth_number: 9,
    tenth_number: 10,
    king_number: 6,
    lady_number: 4,
    jack_number: 2
}

def create_cards():
    cards = {}
```

```
for suit in suits:
    for number in numbers:
        card = Card(suit, number, weights[number])
        cards[card.id] = card
return cards
```



## card.py

```
class Card:
    def __init__(self, suit, number, weight):
        self.id = suit + "___" + number
        self.suit = suit
        self.number = number
        self.weight = weight
        if self.suit == "Clubs":
            self.directory = "cards/" + self.suit + "/" + self.number +
".png"

    def print_card_name(self):
        return self.suit + " " + self.number

# def dir_to_card_picture(self):
#     if self.suit == "Clubs":
#         directory = "cards/" + self.suit + "/" + self.number +
".JPG"
#         return directory
```

## gamer.py

```
MAX_CARD_COUNT = 5
```

```
class Gamer:
    def __init__(self):
        self.cards = []

    def print_cards(self, **args):
        string = ''
        number = 0
        mode = args.get('first_card')
        if mode:
            string += "1: " + self.cards[0].print_card_name() + "\n"
            return string
        for card in self.cards:
            number += 1
            string += str(number) + ": " + card.print_card_name() +
"\n"
        return string
        # dirs = []
        # for card in self.cards:
        #     dirs.append(card.dir_to_card_picture())
        # return dirs

    def add_new_card(self, card):
        if self.count_cards() < 5:
            self.cards.append(card)

    def sum(self):
        sum = 0
        for card in self.cards:
            sum += card.weight
        return sum

    def count_cards(self):
        return len(self.cards)

#bot = Gamer()
#client = Gamer()

#bot.add_new_card()
#client.add_new_card(Card(clubs_suit, a_number, weights[a_number]))
```

## game.py

```
from gamer import Gamer
from create_cards import create_cards
import random

class Game:
    def __init__(self):
        self.all_cards = create_cards()
        self.bot = Gamer()
        self.client = Gamer()
        self.init_first_cards()

    def get_available_cards(self):
        cards_map = self.all_cards.copy()
        for card in self.bot.cards:
            del cards_map[card.id]
        for card in self.client.cards:
            del cards_map[card.id]
        return cards_map

    def init_first_cards(self):
        for i in range(0, 2):
            available_cards = self.get_available_cards()
            available_cards_keys = list(available_cards.keys())
            rand_id = random.choice(available_cards_keys)
            self.bot.add_new_card(available_cards[rand_id])

        for i in range(0, 2):
            available_cards = self.get_available_cards()
            available_cards_keys = list(available_cards.keys())
            rand_id = random.choice(available_cards_keys)
            self.client.add_new_card(available_cards[rand_id])

    def add_new_cards_to_bot(self):
        available_cards = self.get_available_cards()
        available_cards_keys = list(available_cards.keys())
        rand_id = random.choice(available_cards_keys)
        self.bot.add_new_card(available_cards[rand_id])

    def add_new_cards_to_client(self):
        available_cards = self.get_available_cards()
        available_cards_keys = list(available_cards.keys())
        rand_id = random.choice(available_cards_keys)
        self.client.add_new_card(available_cards[rand_id])
```

Да	Нет
<p>Ваши карты:</p> <p>1: Буби 6</p> <p>2: Буби 3</p> <p>3: Пики Валет</p> <p>4: Буби 2</p>	
Очки: 13	
Победитель client !	
<hr/>	
ИТОГИ:	
Ваши карты:	
1: Буби 6	
2: Буби 3	
3: Пики Валет	
4: Буби 2	
Очки: 13	
Карты бота:	
1: Крести 10	
2: Черви 6	
3: Пики Дама	
4: Буби 7	
Очки: 27	
<hr/>	
До свидания!	
Будем рады видеть Вас снова	

Нажатие кнопки “Нет”

 <div><div>Nastya Ищенко</div><div>/start</div></div>	<div>✓5:20</div>
<div><div>N</div><div>Nastino 777</div><div>Добро пожаловать в наше заведение!</div></div>	<div>5:20</div>
<div><div>N</div><div>Nastino 777</div><div>Сыграете в блэкджек?</div></div> <div><div>Согласиться</div><div>Отказаться</div></div>	<div>5:20</div>
<div>Ваши карты: 1: Черви Король 2: Черви Валет</div>	<div>5:20</div>
<div>Очки: 8</div>	<div>5:20</div>
<div>Карта бота: 1: Крести 9</div>	<div>5:20</div>
<div><div>N</div><div>Nastino 777</div><div>Еще?</div></div> <div><div>Да</div><div>Нет</div></div>	<div>5:20</div>
<div>Победитель бот :{</div>	<div>5:21</div>
<div>ИТОГИ:</div>	<div>5:21</div>
<div>Ваши карты: 1: Черви Король 2: Черви Валет</div>	<div>5:21</div>
<div>Очки: 8</div>	<div>5:21</div>
<div>Карты бота: 1: Крести 9 2: Крести Туз</div>	<div>5:21</div>
<div>Очки: 20</div>	<div>5:21</div>
<div>До свидания! Будем рады видеть Вас снова</div>	<div>5:21</div>