МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №8 по дисциплине

«Разработка программных систем»

Выполнил студент группы ИВТ-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Седов М. Д./

Проверил доцент кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Долженкова М. Л./

Киров 2020

1 Задание

В ходе выполнения лабораторной работы необходимо решить следующие задачи:

* Согласовать тематику разработки с преподавателем.
* Разработать структуры серверного и клиентского приложений.
* Реализовать приложение.
* Продемонстрировать работу приложения.

2 Листинг программы

Листинг программы приведен в приложении А.

1. Вывод

В ходе выполнения лабораторной работы был изучен основные возможности протокола XML-RPC. Разработано серверное приложение для генерации простых случайных чисел и операции над ними, для взаимодействия с которыми используется протокол XML-RPC. На основе предыдущей работы разработано клиентское приложение, делегирующее выполнение вычислительных действий серверному приложению. Для организации взаимосвязи между приложениями был использовал модель xmlrpc.

Приложение А

(обязательное) Листинг программы

**server.py**

from xmlrpc.server import SimpleXMLRPCServer

from prime\_numbers import is\_prime, is\_pairwise\_primes, is\_relative\_primes, get\_next, get\_prev, get\_random\_prime\_array, get\_random\_prime\_number

def main():

    server = SimpleXMLRPCServer(("127.0.0.1", 8080))

    server.register\_function(is\_prime)

    server.register\_function(is\_pairwise\_primes)

    server.register\_function(is\_relative\_primes)

    server.register\_function(get\_next)

    server.register\_function(get\_prev)

    server.register\_function(get\_random\_prime\_array)

    server.register\_function(get\_random\_prime\_number)

    server.serve\_forever()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**main.py**

import tkinter as tk

import prime\_numbers as pn

from tkinter import messagebox

from xmlrpc.client import ServerProxy

MIN\_VALUE = 1

MAX\_VALUE = 10\*\*9

def read\_int(string, min\_value, max\_value):

try:

value = int(string)

if min\_value <= value <= max\_value:

return value

return None

except ValueError:

return None

def read\_int\_array(s, min\_value, max\_value):

values = s.split(' ')

numbers = []

for i in values:

v = read\_int(i, min\_value, max\_value)

if v is None:

return None

else:

numbers.append(v)

return numbers

class ReadOnlyText(tk.Text):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self['state'] = tk.DISABLED

def set\_text(self, text: str):

self['state'] = tk.NORMAL

self.delete('0.0', tk.END)

self.insert('0.0', text)

self['state'] = tk.DISABLED

class MainFrame(tk.Frame):

def \_\_init\_\_(self, server, parent=None):

super().\_\_init\_\_(master=parent)

self.\_server = server

self.\_build\_ui()

def \_build\_ui(self):

rbuttons\_frame = tk.Frame(self)

type\_label = tk.Label(rbuttons\_frame, text='Выбор команды:')

type\_label.pack(side=tk.TOP)

self.\_type\_var = tk.StringVar()

self.\_op\_code = tk.StringVar()

self.\_op\_code = "3"

self.random\_array = tk.Radiobutton(rbuttons\_frame, text='Случайный массив', variable=self.\_type\_var, value='random\_array', command=self.change\_name\_method)

self.random\_array.select()

self.random\_array.pack(side=tk.TOP)

self.random\_prime\_mn = tk.Radiobutton(rbuttons\_frame, text='Случайное число (m, n)', variable=self.\_type\_var, value='random\_prime\_mn', command=self.change\_name\_method)

self.random\_prime\_mn.pack(side=tk.TOP)

self.is\_relative = tk.Radiobutton(rbuttons\_frame, text='Взаимно-простые числа', variable=self.\_type\_var, value='is\_relative', command=self.change\_name\_method)

self.is\_relative.pack(side=tk.TOP)

self.is\_pairwise = tk.Radiobutton(rbuttons\_frame, text='Попарно-простые числа', variable=self.\_type\_var, value='is\_pairwise', command=self.change\_name\_method)

self.is\_pairwise.pack(side=tk.TOP)

self.get\_next = tk.Radiobutton(rbuttons\_frame, text='Следующее простое число', variable=self.\_type\_var, value='get\_next', command=self.change\_name\_method)

self.get\_next.pack(side=tk.TOP)

self.get\_prev = tk.Radiobutton(rbuttons\_frame, text='Предыдущее простое число', variable=self.\_type\_var, value='get\_prev', command=self.change\_name\_method)

self.get\_prev.pack(side=tk.TOP)

self.is\_prime = tk.Radiobutton(rbuttons\_frame, text='Проверка на простоту', variable=self.\_type\_var, value='is\_prime', command=self.change\_name\_method)

self.is\_prime.pack(side=tk.TOP)

rbuttons\_frame.pack(side=tk.LEFT, fill=tk.X)

#params frame

params\_frame = tk.Frame(self)

self.name\_method\_label = tk.Label(params\_frame, text='Случайный массив', width=25)

self.name\_method\_label.pack(side=tk.TOP)

self.\_command\_var = tk.StringVar()

self.command\_entry = tk.Entry(params\_frame, textvariable=self.\_command\_var, width=20)

self.command\_entry.pack(side=tk.TOP)

execute\_command\_button = tk.Button(params\_frame, text='Выполнить команду', command=self.\_execute\_command)

execute\_command\_button.pack(side=tk.TOP)

params\_frame.pack(side=tk.LEFT)

#result frame by command

result\_frame = tk.Frame(self)

scrollbar = tk.Scrollbar(result\_frame)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

self.\_result\_text = ReadOnlyText(result\_frame, yscrollcommand=scrollbar.set)

scrollbar['command'] = self.\_result\_text.yview

self.\_result\_text.pack(side=tk.LEFT)

result\_frame.pack(side=tk.TOP)

def \_execute\_command(self):

s = self.\_command\_var.get()

op\_code = int(self.\_op\_code)

if op\_code == 2:

l = read\_int\_array(s, MIN\_VALUE, MAX\_VALUE)

if l is None or len(l) != 2 or l[0] > l[1]:

messagebox.showerror('Error', 'Wrong numbers')

else:

m, n = l[0], l[1]

self.\_result\_text.set\_text(self.\_server.get\_random\_prime\_number(m, n))

elif op\_code == 3:

value = read\_int(s, MIN\_VALUE, MAX\_VALUE)

if value is None:

messagebox.showerror('Error', 'Wrong length of array')

else:

array = self.\_server.get\_random\_prime\_array(value)

self.\_result\_text.set\_text(self.\_dump\_json(array))

elif op\_code == 4:

l = read\_int\_array(s, MIN\_VALUE, MAX\_VALUE)

if l is None:

messagebox.showerror('Error', 'Wrong array')

else:

self.\_result\_text.set\_text(self.\_server.is\_relative\_primes(l))

elif op\_code == 5:

l = read\_int\_array(s, MIN\_VALUE, MAX\_VALUE)

if l is None:

messagebox.showerror('Error', 'Wrong array')

else:

self.\_result\_text.set\_text(self.\_server.is\_pairwise\_primes(l))

elif op\_code == 6:

value = read\_int(s, MIN\_VALUE, MAX\_VALUE)

if value is None:

messagebox.showerror('Error', 'Wrong number')

else:

self.\_result\_text.set\_text(self.\_server.get\_next(value, MAX\_VALUE))

elif op\_code == 7:

value = read\_int(s, MIN\_VALUE, MAX\_VALUE)

if value is None:

messagebox.showerror('Error', 'Wrong number')

else:

self.\_result\_text.set\_text(self.\_server.get\_prev(value, MIN\_VALUE))

elif op\_code == 8:

value = read\_int(s, MIN\_VALUE, MAX\_VALUE)

if value is None:

messagebox.showerror('Error', 'Value must be in range(1, 10^9)')

else:

self.\_result\_text.set\_text(self.\_server.is\_prime(value))

@staticmethod

def \_dump\_json(array):

from json import dumps

return dumps(array, indent=2)

def change\_name\_method(self):

self.command\_entry.delete(0, 'end')

if self.\_type\_var.get() == "random\_prime\_mn":

self.name\_method\_label['text'] = self.random\_prime\_mn['text']

self.\_op\_code = "2"

elif self.\_type\_var.get() == "random\_array":

self.name\_method\_label['text'] = self.random\_array['text']

self.\_op\_code = "3"

elif self.\_type\_var.get() == "is\_relative":

self.name\_method\_label['text'] = self.is\_relative['text']

self.\_op\_code = "4"

elif self.\_type\_var.get() == "is\_pairwise":

self.name\_method\_label['text'] = self.is\_pairwise['text']

self.\_op\_code = "5"

elif self.\_type\_var.get() == "get\_next":

self.name\_method\_label['text'] = self.get\_next['text']

self.\_op\_code = "6"

elif self.\_type\_var.get() == "get\_prev":

self.name\_method\_label['text'] = self.get\_prev['text']

self.\_op\_code = "7"

elif self.\_type\_var.get() == "is\_prime":

self.name\_method\_label['text'] = self.is\_prime['text']

self.\_op\_code = "8"

class NumberFrame(tk.Frame):

def \_\_init\_\_(self, server, parent=None):

super().\_\_init\_\_(master=parent)

self.\_server = server

self.\_build\_ui()

def \_build\_ui(self):

gen\_number\_bt = tk.Button(self, text='Случайное простое число', command=self.\_gen\_number)

gen\_number\_bt.pack(side=tk.LEFT)

self.\_number\_var = tk.StringVar()

number\_entry = tk.Entry(self, textvariable=self.\_number\_var, state='readonly')

number\_entry.pack(side=tk.LEFT)

def \_gen\_number(self):

self.\_number\_var.set(self.\_server.get\_random\_prime\_number(2, pn.MAX\_NUMBER))

class Application(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.\_server = ServerProxy('http://127.0.0.1:8080')

self.\_build\_ui()

def \_build\_ui(self):

self.title('Prime numbers')

self.\_number\_frame = NumberFrame(self.\_server, self)

self.\_number\_frame.pack(side=tk.TOP)

self.\_main\_frame = MainFrame(self.\_server, self)

self.\_main\_frame.pack(side=tk.TOP)

if \_\_name\_\_ == "\_\_main\_\_":

app = Application()

app.mainloop()