МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Лабораторная работа №8 по курсу «Разработка программных систем»

Выполнил студент группы ИВТ-32 _____ /Рзаев А. Э./
Проверил доцент кафедры ЭВМ _____/Чистяков Г. А./

Киров 2017

1 Задание

В ходе выполнения лабораторной работы необходимо решить следующие задачи:

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

2 Листинг программы

Листинг программы приведен в приложении А.

3 Вывод

В ходе выполнения лабораторной работы были изучены основные возможности протокола XML-RPC. Разработано серверное приложение для генерации простых случайных чисел, для взаимодействия с которым используется протокол XML-RPC. На основе предыдущей работы разработано клиентское приложение, делегирующее выполнение вычислительных действий серверному приложению. Для организации взаимосвязи между приложениями был использован модуль xmlrpc.

Приложение А
(обязательное)
Листинг программы

server.py

```
from xmlrpc.server import SimpleXMLRPCServer
from primerandom import get_random_array, get_random_number

def main():
    server = SimpleXMLRPCServer(("127.0.0.1", 8080))
    server.register_function(get_random_number)
    server.register_function(get_random_array)
    server.serve_forever()

if __name__ == '__main__':
    main()
```

client.py

```
import tkinter as tk
from tkinter import messagebox
from xmlrpc.client import ServerProxy

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 ArrayFrame(tk.Frame):
    def __init__(self, server, parent=None):
        super().__init__(master=parent)
        self._server = server
        self._build_ui()

    def _build_ui(self):
        params_frame = tk.Frame(self)
        array_len_label = tk.Label(params_frame, text='Length of array')
        array_len_label.pack(side=tk.LEFT)
        self._array_len_var = tk.StringVar()
        array_len_entry = tk.Entry(params_frame, textvariable=self._array_len_var,
            width=20)

        array_len_entry.pack(side=tk.LEFT)
        gen_array_button = tk.Button(params_frame, text='Get array', command=self.
           _gen_array)
        gen_array_button.pack(side=tk.LEFT)
        params_frame.pack(side=tk.TOP)

        type_frame = tk.Frame(self)
        type_label = tk.Label(type_frame, text='Data format:')
        type_label.pack(side=tk.LEFT)

        self._type_var = tk.StringVar()
```

```
        type_json = tk.Radiobutton(type_frame, text='JSON', variable=self.
            _type_var, value='JSON')
        type_json.select()
        type_json.pack(side=tk.LEFT)

        type_xml = tk.Radiobutton(type_frame, text='XML', variable=self._type_var,
             value='XML')
        type_xml.pack(side=tk.LEFT)

        type_frame.pack(side=tk.TOP)

        array_frame = tk.Frame(self)
        scrollbar = tk.Scrollbar(array_frame)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self._array_text = ReadOnlyText(array_frame, yscrollcommand=scrollbar.set)
        scrollbar['command'] = self._array_text.yview
        self._array_text.pack(side=tk.LEFT)
        array_frame.pack(side=tk.TOP)

    def _gen_array(self):
        length = self._array_len_var.get()
        if length.strip().isdecimal() and 1 <= int(length) <= 100000:
            array = self._server.get_random_array(int(length))
            if self._type_var.get() == 'JSON':
                self._array_text.set_text(self._dump_json(array))
            else:
                self._array_text.set_text(self._dump_xml(array))
        else:
            messagebox.showerror('Error', 'Length of array must be in [1; 100000]'
                )

    @staticmethod
    def _dump_json(array):
        from json import dumps
        return dumps(array, indent=2)

    @staticmethod
    def _dump_xml(array):
        return '<?xml version="1.0" ?>\n' +\
               '<array>\n' +\
               '\n'.join('  <item>{}</item>'.format(v) for v in array) +\
               '\n</array>'


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_button = tk.Button(self, text='Get prime random number',
            command=self._gen_number)
        gen_number_button.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_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 random number generator')
        self._number_frame = NumberFrame(self._server, self)
        self._number_frame.pack(side=tk.TOP)
        self._array_frame = ArrayFrame(self._server, self)
        self._array_frame.pack(side=tk.TOP)

if __name__ == '__main__':
    app = Application()
    app.mainloop()
```