ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
СӘТБАЕВ УНИВЕРСИТЕТІ

Автоматика және Информациялық Технологиялар институты
Программалық Инженерия кафедрасы

SATBAYEV
UNIVERSITY

ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС #6

Тақырыбы: Функционалды реактивті бағдарламалау (FRP)

№	Жұмысты орындау сапасы	Баға диапазоны	Орындалған %
1	Орындалған жоқ	0%	
2	Орындалды	0-50%	
3	Материялдық өзіндік жүйелендіру	0-10%	
4	Талап етілген көлемде және көрсетілген мерзімде орындау	0-5%	
5	Қосымша ғылыми әдебиеттерді пайдалану	0-5%	
6	Орындаған тапсырманың ерекшелігі	0-10%	
7	СӨЖ-ді қорғау	0-20%	
	Қорытынды:	0-100%	

Оқытушы: Шаяхметов Д
Студент: Ұлдақан А
Мамандығы: Computer Science
Тобы: Дс 7:50 – 9:45

Алматы 2024 ж

Мақсат:
Python контекстінде функционалдық реактивті бағдарламалау (FRP) принциптерін үйреніңіз және қолданыңыз. Жұмыстың мақсаты - деректер ағындарын, реактивті айнымалыларды және асинхронды оқиғаларды өңдеуді қоса алғанда, FRP негізгі тұжырымдамаларын түсіну және жауап беретін және модульдік қосымшаларды жасау үшін осы тұжырымдамаларды пайдалану дағдыларын дамыту.
Тапсырмалар:
1. FRP негіздері:
- Ағындар, сигналдар және реактивті айнымалылар сияқты негізгі FRP тұжырымдамаларын үйреніңіз.
- реактивті өзгерістерді тарату және тәуелділікті басқару механизмдерін талдау.
2. Қарапайым FRP сценарийлерін іске асыру:
- өзгермелі деректерді өңдеу және көрсетумен байланысты мәселелерді шешу үшін FRP қолдану.
- Пайдаланушы әрекеттеріне немесе сыртқы оқиғаларға жауап ретінде реактивті деректерді жаңартуды көрсететін мысалдарды әзірлеу.
3. Қолданыстағы қолданбалармен интеграция:
- Қолданыстағы Python қолданбаларына FRP тәсілдерін біріктіру жолдарын қарастыру.
- FRP қолдану арқылы жауап беруді жақсарту және код күрделілігін азайту мүмкіндіктерін талдау.
4. FRP-дегі қосымша тақырыптар:
- Қателерді өңдеу, ағынды біріктіру және асинхронды операцияларды қоса алғанда, FRP жүйесіндегі озық әдістер мен үлгілерді үйреніңіз.
- көптеген деректер көздерін тиімді өңдеуге қабілетті күрделі реактивті жүйелерді жасау.
5. Сыни талдау және рефлексия:
- Нақты әлемдегі қолданбалар контекстінде FRP артықшылықтары мен шектеулерін бағалау.
- Нақты бағдарламалық тапсырмалар үшін тәсілдер мен құралдарды таңдауда сыни тұрғыдан ойлауды дамыту.
Зертханалық жұмыстың маңыздылығы:
Зертхана студенттерді қазіргі заманғы бағдарламалық қамтамасыз етуді әзірлеуде маңыздырақ болып келе жатқан функционалдық реактивті бағдарламалау парадигмасымен таныстыруға арналған. FRP интерактивті және асинхронды қолданбаларды құруға арналған талғампаз шешімдерді ұсынады, бұл күй мен деректер ағынын басқаруды жеңілдетеді. Жұмыс реактивті жүйелерді терең түсінуді дамытады және анағұрлым жауап беретін және модульдік қосымшаларды жобалау дағдыларын жетілдіреді.

Жеке тапсырмалар:
Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз).

2	2021-2022	Бак	6B06102 Computer Science	Ұлдақан Ален Серужанұлы	Полная

Функционалды реактивті бағдарламалау (FRP) өзгерістерге жауап беретін жүйелерді құру үшін реактивті және функционалды бағдарламалау идеяларын біріктіреді.

2. FRP істер тізімі
- Тапсырмаларды қосуға және жоюға жауап беретін FRP көмегімен қарапайым істер тізімі қолданбасын жасаңыз.

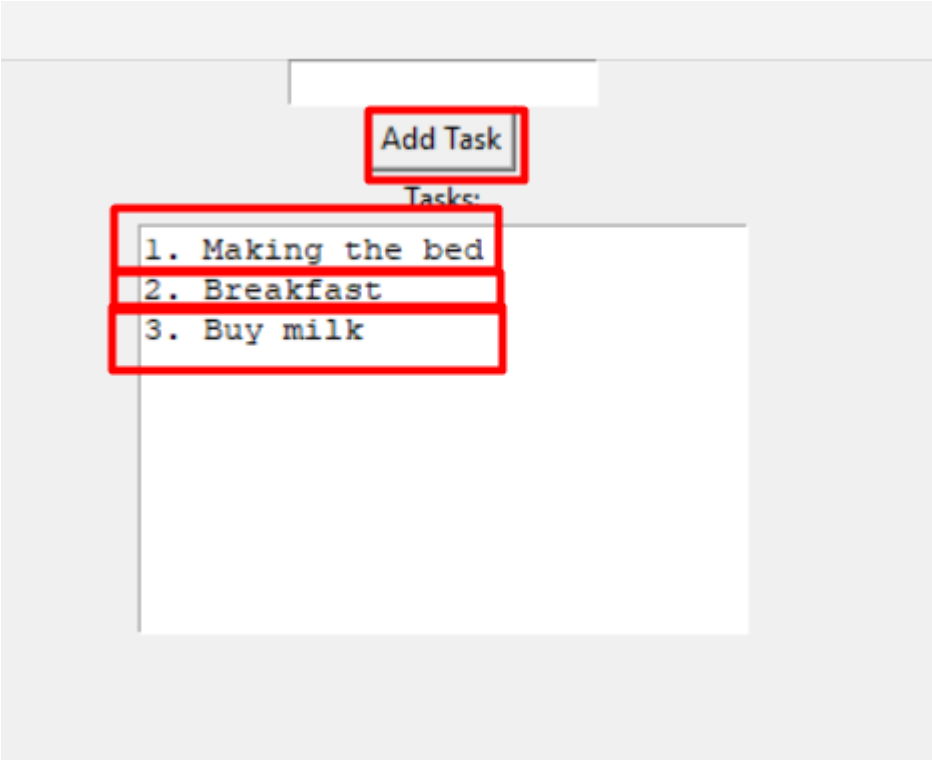
Code:

```
import tkinter as tk
from rx import subject

class TodoList:
    def __init__(self):
        self.tasks = []
        self.subject = subject.Subject()

    def add_task(self, task):
        self.tasks.append(task)
        self.subject.on_next(self.tasks)

    def remove_task(self, task_index):
        del self.tasks[task_index]
        self.subject.on_next(self.tasks)

    def get_tasks(self):
        return self.subject

class TodoApp:
    def __init__(self, todo_list, root):
        self.todo_list = todo_list
        self.root = root
        self.setup_ui()

    def setup_ui(self):
```

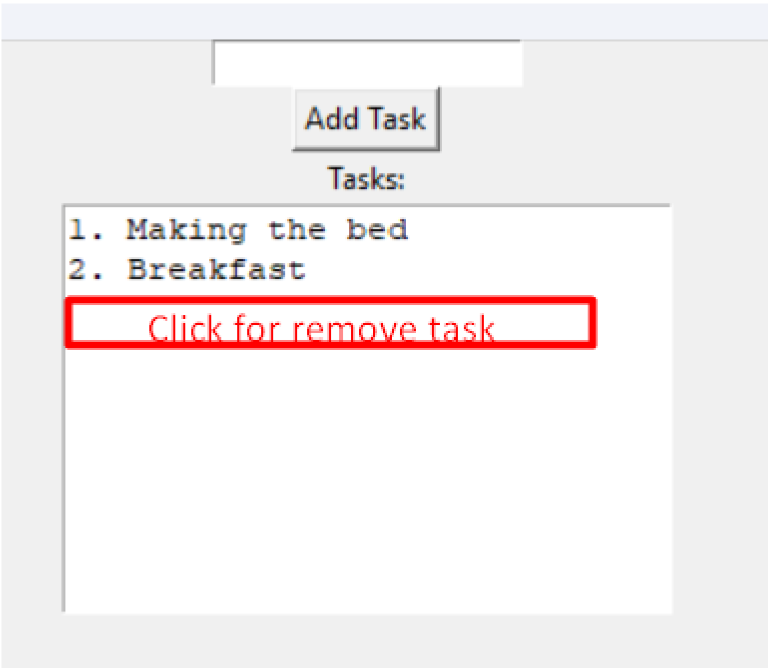
```
        self.task_entry = tk.Entry(self.root)
        self.task_entry.pack()

        self.add_button = tk.Button(self.root, text="Add Task",
command=self.add_task)
        self.add_button.pack()

        self.tasks_label = tk.Label(self.root, text="Tasks:")
        self.tasks_label.pack()

        self.tasks_text = tk.Text(self.root, height=10, width=30)
        self.tasks_text.pack()

        self.task_click_subject = subject.Subject()

        self.tasks_text.bind("<Button-1>", self.on_task_click)

        self.todo_list.get_tasks().subscribe(self.update_ui)

    def update_ui(self, tasks):
        self.tasks_text.delete('1.0', tk.END)
        for i, task in enumerate(tasks):
            self.tasks_text.insert(tk.END, f"{i + 1}. {task}\n")

    def add_task(self):
        task = self.task_entry.get()
        if task:
            self.todo_list.add_task(task)
            self.task_entry.delete(0, tk.END)

    def on_task_click(self, event):
        index = self.tasks_text.index(tk.CURRENT)
        task_index = int(index.split('.')[0]) - 1
        self.task_click_subject.on_next(task_index)

if __name__ == "__main__":
    todo_list = TodoList()
    root = tk.Tk()
    todo_app = TodoApp(todo_list, root)

    todo_app.task_click_subject.subscribe(lambda task_index:
todo_list.remove_task(task_index))

    root.mainloop()
```

Output:

Add tasks:



Remove tasks:



Work with GIT

1. Initialize git repository

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6
$ git init
Reinitialized existing Git repository in C:/Users/Admin/OneDrive/╨Ф╨╛╨║╤Г╨╝╨╡╨╜╤B
gramming/Lab 6/.git/
```

2. Add remote my course repository (for my example GitHUB)

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (master)
$ git remote add origin https://github.com/JackOptimist/Functional-Programming.git
```

3. Create new branch and switched to a new branch

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (master)
$ git checkout -b Lab-6
Switched to a new branch 'Lab-6'
```

4. Our python file is untracked

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (Lab-6)
$ git status
On branch Lab-6

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        to-do_list.py

nothing added to commit but untracked files present (use "git add" to track)
```

5. Now our file is tracked

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (Lab-6)
$ git status
On branch Lab-6

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   to-do_list.py
```

6. Commit

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (Lab-6)
$ git commit -m "First commit"
[Lab-6 (root-commit) 302e22c] First commit
 1 file changed, 68 insertions(+)
 create mode 100644 to-do_list.py
```

7. I pushed files to remote repository (correct branch Lab-6)

```
Admin@Sam MINGW64 ~/OneDrive/Документы/6th semester/Functional Programming/Lab 6 (Lab-6)
$ git push origin Lab-6
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 868 bytes | 434.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'Lab-6' on GitHub by visiting:
remote:      https://github.com/JackOptimist/Functional-Programming/pull/new/Lab-6
remote:
To https://github.com/JackOptimist/Functional-Programming.git
 * [new branch]      Lab-6 -> Lab-6
```

8. Remote repository – GitHUB (My GitHUB Profile)