به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه هفتم

آشنایی با کار با فایل در جاوا

مقدمه

هدف این جلسه، تمرین مباحث مربوط به فایل در جاوا است. این مفاهیم را احتمالا در برنامه‌های مختلف مشاهده کرده‌اید. اینکه چگونه نام کاربری و رمز عبور در یک برنامه ذخیره می‌شود؛ چگونه از اطلاعات پشتیبان[1] بگیریم؛ نمونه‌هایی است که احتمالا با آن مواجه شده‌اید. در این جلسه قصد داریم با استفاده از مفاهیم ذخیره و بازیابی اطلاعات، برنامه‌ای برای ثبت یادداشت‌های روزانه پیاده‌سازی کنیم. لازم به ذکر است در ترم‌های آتی و در درس‌های پایگاه داده و ذخیره و بازیابی اطلاعات، به طور دقیق‌تر با روش‌های ذخیره‌سازی و استفاده از اطلاعات آشنا خواهید شد و در جلسه صرفا با روش‌های مقدماتی آشنا خواهید شد.

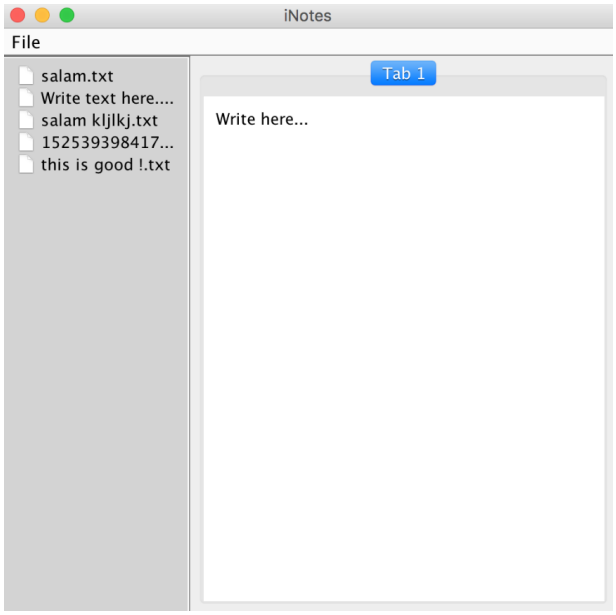
نکات آموزشی

همان‌طور که در مطالب درسی به آن اشاره شد برای بازیابی و ذخیره فایل‌ها می‌توان از روش‌های مختلفی استفاده کرد. استفاده از کلاس‌های FileWriter/FileReader یا FileInputStream/FileOutputStream (گاهی اوقات به همراه BufferedReader/BufferedWriter) روش‌های متداولی است که اغلب مورد استفاده قرار می‌گیرند. همچنین گاهی اوقات با استفاده از مفهوم سریال‌سازی[2] می‌توان فرآیند ذخیره‌سازی را تسهیل کرد و مستقیما اشیا را روی فایل ذخیره و بازیابی کرد. در ادامه با این روش‌ها آشنا خواهیم شد.

[1] backup
[2] serialization

مراحل انجام کار

در این پروژه، قصد داریم به صورت مرحله به مرحله برنامه‌ای بنویسیم که یک اپلیکشن یادداشت-برداری را شبیه‌سازی کند. همان‌طور که می‌دانید یک برنامه یادداشت‌برداری در ساده‌ترین حالت می‌تواند نوشته‌های فرد را ذخیره کند و هنگام باز شدن برنامه، آنها را به وی نشان دهد.



شکل ۰۱-انمونه‌ای از واسط گرافیکی برنامه یادداشت برداری

انجام دهید:

برنامه‌ی یادداشت برداری را به نحوی طراحی و پیاده‌سازی کنید که:

۱. امکان ذخیره‌ی یادداشت‌های جدید برای کاربر فراهم باشد.
۲. کاربر قادر باشد خلاصه‌ای از تمام یادداشت‌های خود را مشاهده کند.
۳. کاربر بتواند یادداشت دلخواهی را انتخاب کرده و متن کامل آن را مشاهده کند.
٤. امکان حذف هرکدام از یادداشت‌ها برای کاربر فراهم باشد.

قسمت‌های مربوط به خواندن و نوشتن روی فایل تنها در کلاس FileUtils و به دو صورت پیاده‌سازی شوند (یکبار با کمک BufferedWriter/BufferedReader و بار دیگر به کمک FileInputStream/FileOutputStream).

٥. همانطور که می‌دانید، می‌توان از برنامه‌های جاوا فایل اجرایی ساخت و مانند برنامه‌های exe آن را اجرا کرد (با شرط وجود JRE). پس از تکمیل قسمت‌های فوق با استفاده از مسیر زیر در IntelliJ یک فایل jar از پروژه ایجاد کرده و به مدرس کارگاه نشان دهید (دقت کنید

مسیر پیش‌فرض فایل jar در پوشه out است. در حالی که برای اجرای درست برنامه مسیر پیش‌فرض باید پوشه notes باشد!).

File -> Project Structure -> Project Settings -> Artifacts -> Click green plus sign -> Jar -> From modules with dependencies...

The above sets the "skeleton" to where the jar will be saved to. To actually build and save it do the following:

Extract to the target Jar

OK

Build | Build Artifact

شکل ۲-۰۱ نحوه ساخت یک فایل اجرایی در IntelliJ

یک راه حل مناسب برای کاهش پیچیدگی و تسهیل فرآیند ذخیره فایل در برنامه بالا استفاده از مفهوم سریال‌سازی است. سریال‌سازی به این معناست که بایت‌های اشغال شده توسط یک شیئ در حافظه را به صورت پشت سر هم درآوریم. هدف از این کار معمولا ذخیره‌سازی اشیاء بر روی فایل و استفاده مجدد از آن در آینده است. بنابراین برای اینکه بتوان یک شیئ را مستقیم روی فایل ذخیره کرد، باید سریال‌پذیر باشد که این امر با پیاده‌سازی اینترفیس Serializable توسط کلاس مورد نظر انجام می‌گردد. به عنوان مثال برای اینکه در برنامه بالا بخواهیم علاوه بر محتوای یادداشت اطلاعات دیگری را ذخیره کنیم می‌توان یک کلاس مانند Note ساخت که اینترفیس Serializable پیاده‌سازی کرده باشد. در نتیجه می‌توان هر شیئ از ساخته شده از این کلاس را به طور کامل در فایل ذخیره‌سازی کرد.

```java
import java.io.Serializable;

public class Note implements Serializable {

    private String title;
    private String content;
    private String date;

    public Note(String title, String content, String date) {
        this.title = title;
        this.content = content;
        this.date = date;
    }

    public String getTitle() {
```

برای خواندن و نوشتن یک شیئ به کمک Serialization می‌توان به صورت زیر عمل کرد:

```
Note note1 = new Note("Test1", "this is dummy content1", "1397/2/15");
Note note2 = new Note("Test2", "this is dummy content2", "1397/2/15");


try (FileOutputStream fs = new FileOutputStream("note.bin")) {

    ObjectOutputStream os = new ObjectOutputStream(fs);

    os.writeObject(note1);
    os.writeObject(note2);


} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

```
try (FileInputStream fi = new FileInputStream("note.bin")) {

    ObjectInputStream os = new ObjectInputStream(fi);

    Note note1 = (Note) os.readObject();
    Note note2 = (Note) os.readObject();


    System.out.println(note1);
    System.out.println(note2);



} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

همانطور که در شکل بالا مشاهده می‌کنید برای خواندن و نوشتن یک شیئ روی فایل از کلاس ObjectOutputStream/ObjectInputStream استفاده شده است و برخلاف قبل که از توابع write/read استفاده می‌شد در اینجا از توابع writeObject/readObject استفاده می‌کنیم.

نکته: همانطور که در شکل بالا مشاهده می‌شود نحوهٔ نوشتن try/catch اندکی متفاوت است. در این روش که اصطلاحا به آن try-with-resources گفته می‌شود دیگر نیازی به فراخوانی تابع بستن (close) نیست. این ساختار بعد از اتمام عبارت، تابع close را به طور خودکار فراخوانی می‌کند.

انجام دهید:

پیاده‌سازی خود را به نحوی تغییر دهید که علاوه بر محتوای یادداشت، تاریخ و عنوان آن (نام فایل) ذخیره شود. همچنین لیست فایل‌ها را به نحوی تغییر دهید که علاوه بر خط اول یادداشت، تاریخ و عنوان آن به کاربر نمایش داده شود. دقت داشته باشید موارد مربوط به خواندن و نوشتن فایل باید تنها در FileUtils انجام شود. در این بخش باید از مفهوم سریال‌سازی برای ذخیره‌ی داده‌ها استفاده شود.