*Eack Xoa

سینا قاسمی‌نژاد

فروردین ۱۴۰۰

این فایل برای توضیح فاز اول پروژه درس برنامه‌نویسی پیشرفته تهیه شده است.

کد

در این بخش به توضیح پکیج‌ها و کلاس‌های پروژه می‌پردازم.

0.0 - Main

کلاس مین در ابتدای کار استارت شده و تنها کاری که می‌کند این است که کاربر را به کلاس FirstPage منتقل کند.

1 - utils

در پکیج utils کدهای کمکی برای اجرای درست و بهتر پروژه قرار گرفته اند.

1.1 - ConsoleColors

برای رنگی‌کردن خروجی کنسول. کد این بخش در داک پروژه قرار داشت.

1.2 - Input

کلاس Input برای جلوگیری از تعریف تعداد زیادی اسکنر در سطح کد و کثیف شدن آن و اجتناب از گرفتن تعداد زیادی ارور NoSuchElementException به‌دلیل بسته شدن اسکنر و تعریف اسکنر جدید در کلاسی دیگر ایجاد شده است. در این کلاس یک اسکنر ساخته شده و در مواقع لزوم به کلاس‌های دیگر داده می‌شود.

1.3 - Validations

در هنگام ثبت‌نام، خیلی از کاربران از وارد کردن عبارتی عجیب مثل jnvksdvnk به جای ایمیل یا شماره خود استفاده می‌کنند. بعضی دیگر هم به اشتباه ورودی‌ای به برنامه می‌دهند که ممکن است منجر به باگ خوردن برنامه شود. همچنین برخی دیگر از انها یوزرنیم یا ایمیل یا موبایلی وارد می‌کنند که قبلا استفاده شده است. کاربرد کلاس Validations این است که این قبیل مشکلات را هندل می‌کند. برای چک کردن درست بودن ورودی از RegEx و برای بخش چک کردن تکراری بودن ورودی‌ها از گشتن در فایل‌های اضافه‌ای که بعدتر توضیح داده می‌شود استفاده شده است.

*ایاک سوا

1.4 - DataStructuresUtil

در کلاس MapUtil تابعی نوشته شده است که یک مپ بعنوان ورودی می‌گیرد و در خروجی این مپ را به‌صورت سورت شده نسبت به مقدار value دوتایی‌ها چاپ می‌کند. از این کلاس در Timeline برای مرتب‌سازی توییت‌ها بر طبق زمان توییت شدنشان استفاده شده است.

کد مربوط به این تابع را در لینک زیر پیدا کردم:

```
https://cutt.ly/ixNiCNI
```

تابعی دیگر در این کلاس دو اری لیست بعنوان ورودی گرفته و اجتماع آن‌ها را در خروجی می‌دهد. از این تابع برای share کردن توییت هنگامی که یک نفر در بیش از یک گروه حضور داشته باشد استفاده می‌شود.

کد مربوط به این تابع را در لینک زیر پیدا کردم:

```
https://cutt.ly/FxNiPGN
```

1.5 - UsersCli, TweetsCli, NotificationsCli, SearchUsersCli

برای صفحه‌بندی و نمایش مرتب‌شده کاربران (در بخش فالوورها و فالوویِنگ‌های هر کاربر)، توییت‌ها (در تایم‌لاین، صفحه شخصی هر کاربر و بخش توییت‌های رندوم بخش اکسپلور)، نوتیفیکیشن‌ها و نتایج بدست‌آمده از سرچ نام یا یوزرنیم یک کاربر دیگر در CLI از این کلاس‌ها استفاده شده است. در آنها ابتدا با توجه به پارامتر perPage مورد دلخواه هر کاربر اشیا مربوطه به هاییArrayList مجزا بخش‌بندی شده که هر بخش یک صفحه را نشان می‌دهد، سپس تابعی برای چاپ اعضای این بخش‌ها ساخته شده است. کاربر نیز در هنگام استفاده از برنامه می‌تواند با دکمه (دستور)های next و previous و همچنین page next و page previous در بین این اشیا حرکت کند.

2 - models

در پکیج models مدل‌های برنامه قرار گرفته اند. درباره آن‌ها به‌طور مفصل توضیح داده خواهد شد.

2.1 - User

در این برنامه، هر کاربر یک شی است. این کلاس فیلدهای مختلفی دارد که به موارد مهم آن می‌پردازیم:

- **id:**

  هر کاربر یک آی‌دی دارد که منحصر بفرد است و وی از ابتدای ثبت‌نام تا انتها این آی‌دی به خصوص را خواهد داشت. آی‌دی تغییر نمی‌کند و روند انتخاب آن به ترتیب ثبت‌نام است. برای فهمیدن این‌که آخرین کسی که ثبت‌نام کرده از چه آی‌دی‌ای برخوردار بوده تا آی‌دی جدید را برای کاربر جدیدمان بسازیم، از فایل id.txt که در بخش فایل‌های اضافه به آن اشاره شده است و تابع changeLastId کلاس Save استفاده شده است.

- **username:**

  مثل یوزرنیم عادی کاربران در شبکه‌های مجازی است. منحصر بفرد ولی قابل تغییر. برای فهمیدن این‌که یک یوزرنیم قبلا استفاده شده است یا خیر، از فایل usernames.txt که در بخش فایل‌های اضافه به آن اشاره شده است و کلاس Validations استفاده شده است. در صورت تغییر یوزرنیم یک کاربر، با استفاده از توابع مربوط به تابع replaceLine موجود در کلاس Save این فایل نیز تغییر می‌کند.

- **name:**

  اسم کاربر

- **password:**

  پسورد کاربر

- **birthDate:**

  تاریخ تولد کاربر که یک شی از کلاس Date جاوا است.

- **email:**

  ایمیل کاربر که منحصر بفرد است. توضیحات آن مانند یوزرنیم می‌باشد.

- **phoneNumber:**

  شماره تماس کاربر که منحصر بفرد است. توضیحات آن مانند یوزرنیم می‌باشد.

- **bio:**

  بیو کاربر

- **lastLogin:**

  تاریخ آخرین ورود کاربر به برنامه که هر بار موقع Login تغییر می‌کند.

- **lastSeen:**

  تاریخ آخرین زمانی که یک کاربر آنلاین بوده است. مورداستفاده در بخش دایرکت.

  برای آپدیت خودکار این متغیر از مهدی سلحشور مشورت گرفتم و او مرا با نتیجه جستجوی گوگل خود که استفاده از کتابخانه Timer بود آشنا کرد.

- **isActive:**

  یک بولین که مشخص می‌کند یک کاربر اکتیو است یا اکانت خود را دی‌اکتیو کرده است.

- **isPermitted:**

  یک بولین که مشخص می‌کند کاربر اجازه توییت زدن دارد یا خیر.

- **followers, followings, blocked, muted, reported, requests, pending:**

  چند لیست که کاملا کارکرد یکسانی دارند و با توجه به اسم کاملا مشخص است چه چیزی را ذخیره می‌کنند. برای جلوگیری از لوپ خوردن برنامه هنگام سیو و لود با Gson ، این لیست‌ها به جای خود کاربران، آی‌دی آن‌ها را ذخیره می‌کنند.

- **newNotifications, oldNotifications:**

  دو لیست که نوتیفیکیشن‌های جدید و قدیم را ذخیره می‌کنند. وقتی کاربر در صفحه نوتیفیکیشن‌های جدید دکمه رفرش یا یک را بزند، تمام نوتیفیکیشن‌های جدیدی که داشته به لیست نوتیفیکیشن‌های قدیمی منتقل می‌شود. نوتیفیکشن‌ها شامل پیام‌های فالو و آنفالو شدن کاربر قبول یا رد شدن درخواست‌های فالوی او به پیج‌های پرایوت می‌باشد.

- **userTweets, retweetedTweets, upvotedTweets, downvotedTweets, savedTweets, reportedTweets:**

  چند لیست که کارکرد کاملا یکسانی دارند. در همه آنها آی‌دی توییت‌های مربوطه به دلیل مشابه لیست‌های کاربران، ذخیره می‌شود.

- **homePageTweets:**

  یک مپ که توییت‌های هر فرد که باید در هوم پیجش نشان داده شوند را نشان می‌دهد. این توییت‌ها عبارتند از توییت‌های خودش که کامنت نبوده و ری‌توییت‌هایی که کرده است. key این مپ یک استرینگ به فرم

  $$a - x - y - n$$

  است که در آن a می‌تواند صفر یا یک باشد. صفر، اگر این توییت ری‌توییت شده بود و یک اگر این توییت مال خود شخص بود. x در حالت دوم کاربردی ندارد ولی در حالت اول، آی‌دی کسی است که توییت را ری‌توییت کرده است. $y - n$ نیز همان آی‌دی توییت است که در آن y آی‌دی شخص توییت‌کننده و n شماره توییتی است که نوشته است.

  value این مپ هم تاریخ نوشته شدن آن به میلی‌ثانیه است. این مپ بعدا توسط کلاس MapUtil طبق این مقادیر سورت می‌شود.

- **lastTweetId:**

  شماره آخرین توییتی که کاربر ارسال کرده است. برای ساخت آی‌دی منحصر بفرد هر توییت استفاده می‌شود.

- **infoState:**

  یک بولین که نشان می‌دهد آیا کاربر می‌خواهد دیگران اطلاعات شخصی (ایمیل، شماره تلفن و تولد) او را ببینند یا خیر.

- **lastSeenState:**

  یک استرینگ که نشان می‌دهد کاربر می‌خواهد وضعیت lastSeen اش را چه کسانی ببینند. هیچکس، همه یا تنها کسانی که فالو می‌کند؟

- **privateState:**

  یک بولین که نشان می‌دهد پیج کاربر پرایوت است یا خیر.

- **reports:**

  تعداد دفعاتی که کاربر ریپورت شده است.

- **reportedUntil:**

  تاریخی که کاربر تا آن زمان محروم از توییت زدن است به میلی‌ثانیه. اگر او ریپورت نشده باشد، این لانگ برابر صفر خواهد بود.

- **tweetsPerPage, peoplePerPage, notificationsPerPage:**

  سه متغیر که نشان می‌دهند کاربر می‌خواهد در هر صفحه در CLI چند توییت، کاربر و نوتیفیکیشن ببیند.

- **directMessages:**

  یک مپ که آیدی هر شخص را به اری‌لیست حاوی پیام‌هایی که بین او و کاربر مبادله شده است وصل می‌کند. در این اری‌لیست چهار نوع استرینگ داریم:

  r:String s:String R:String S:String

  که به ترتیب از چپ به راست، برابرند با: آی‌دی پیامی که دریافت شده است. آی‌دی پیامی که ارسال شده است. آی‌دی توییتی که دریافت شده است. آی‌دی توییتی که ارسال شده است. هر پیام در اری‌لیست یک دوتایی استرینگی است که استرینگ اول شناسه پیام و شناسه دوم نشان‌دهنده خوانده شده یا نشده بودن آن پیام است و با t و f نمایش داده می‌شود.

- **groups:**

  یک مپ که اسم هر گروه را به یک لیست حاوی آی‌دی افراد آن گروه وصل می‌کند.

2.2 - Tweet

هر توییت نیز مانند هر یوزر یک شی است. در ادامه به فیلدهای کلاس Tweet می‌پردازیم:

- **id:**

  آی‌دی هر توییت که منحصر بفرد است. این آیدی به‌طور مفصل در بخش homePageTweets کلاس User توضیح داده شد.

- **owner:**

  یک لانگ که آی‌دی نویسنده توییت را ذخیره می‌کند.

- **text:**

  متن توییت

- **tweetTime:**

  تاریخ نوشته شدن شدن توییت

- **visible:**

  یک بولین که مشخص می‌کند آیا توییت موردنظر دیلیت شده است یا خیر.

- **upperTweet:**

  هر توییت ممکن است خود کامنتی برای یک توییت بالاتر باشد. upperTweet آی‌دی آن توییت را ذخیره می‌کند و در غیر این صورت خالی می‌ماند.

- **upvotes, downvotes, retweets:**

  چند لیست که آی‌دی افرادی که توییت را آپووت، داون‌ووت یا ری‌توییت کرده اند را ذخیره می‌کند.

- **comments:**

  لیست آی‌دی کامنت‌های توییت. دقت کنید که اگر y یک کامنت x باشد و z نیز یک کامنت y انگاه z در لیست کامنت‌های x نخواهد بود.

- **reports:**

  تعداد دفعاتی که این توییت ریپورت شده است.

2.3 - Messages

هر پیام شخصی بین دو فرد یک شی است. این شی خصوصیات زیادی ندارد ولی به آن‌ها می‌پردازیم:

- **id:**

  دقیقا همانند هر User هر پیام نیز یک آی‌دی دارد که به سادگی با توجه به آخرین آی‌دی ساخته شده که در فایل message.txt ذخیره می‌شود ساخته می‌شود.

- **sender, receiver:**

  آی‌دی ارسال‌کننده و دریافت‌کننده پیام

- **text:**

  متن پیام

- **visible:**

  مشخص کننده اینکه یک پیام پاک شده است یا خیر. در این فاز چون خواسته نشده بود این قابلیت اضافه نشد.

- **messageTime:**

  زمان ارسال پیام

3 - entry

در پکیج entry کلاس‌های مربوط به ورود کاربر به برنامه قرار دارند.

3.1 - Enter

اولین محیطی که کاربر با آن روبه‌رو می‌شود و از او پرسیده می‌شود که اکانتی دارد یا خیر. در هر دو صورت به کلاس مربوطه منتقل می‌شود.

3.2 - SignUp

کلاس ثبت‌نام کاربر. در این کلاس از کلاس Validations برای چک کردن ولید بودن یا موجود بودن یوزرنیم، ایمیل، تاریخ تولد و شماره تلفن افراد استفاده شده است. در صورت لزوم و وجود یک اکانت از قبل نیز می‌توان مستقیما از این کلاس به کلاس Login رفت.

3.3 - Login

کلاس ورود کاربر. در صورت دی‌اکتیو بودن اکانت کاربر نیز، در این کلاس می‌تواند آن را اکتیو کند. در صورت لزوم نیز می‌توان مستقیما از این کلاس به کلاس SignUp رفت.

4 - data

این پکیج برای مدیریت فایل‌های کاربران و توییت‌های آن‌هاست.

4.1 - Save

این کلاس به سیو کردن کاربران و توییت‌ها می‌پردازد. همچنین با توابع changeUsername و changeEmail و changePhoneNumber در فایل‌های ذخیره‌سازی لیست یوزرنیم‌ها، ایمیل‌ها و شماره تلفن‌ها گشته و مقدار قبلی را با مقدار جدید جایگزین می‌کند. همچنین هنگام ساخت یک اکانت جدید، با تابع changeLastId مقدار آی‌دی قبلی را پاک کرده و آی‌دی جدید را جایگزین آن می‌کند. پیام‌های شخصی نیز با این کلاس سیو می‌شوند.

4.2 - Load

این کلاس به لود فایل‌های json سیو شده در کلاس Save می‌پردازد. در این کلاس دو تابع برای پیدا کردن یک کاربر توسط یوزرنیم یا آی‌دی وی و همچنین یک تابع برای پیدا کردن توییت وجود دارد. پیام‌ها نیز با این کلاس لود می‌شوند.

5 - pages

صفحات مختلف برنامه در این پکیج وجود دارند.

5.1 - FirstPage

این کلاس صرفا پیام خوش‌آمدگویی برای کاربر چاپ کرده و او را به صفحه ورود و سپس به صفحه اصلی برنامه راهنمایی می‌کند.

5.2 - MainPage

صفحه اصلی برنامه که در آن کاربر می‌تواند به صفحه شخصی خود، تایم‌لاین، تنظیمات، اکسپلور، نوتیفیکیشن‌ها و یا چت دایرکت برود. همچنین می‌تواند برنامه را بسته و یا از اکانت خود بیرون بیاید.

5.3 - HomePage

هوم پیج کاربر. در این صفحه فرد می‌تواند به توییت‌های خودش و لیست فالوورها و فالوویینگ‌ها دسترسی داشته باشد و توییت جدید ارسال کند. تمامی دستورها در صورت وجود به او نمایش داده می‌شوند.

5.3.1 - Followings

لیست فالووینگ‌ها. کاربر در این صفحه می‌تواند وارد صفحه دیگران شده یا آن‌ها را آنفالو یا بلاک و ... کرده و وضعیت فالو شدن خود توسط آن‌ها را نیز ببیند.

5.3.2 - Followers

همانند لیست فالوویینگ‌ها. با قابلیت فالو و آنفالو برای کاربر.

5.3.3 - Blacklist

لیست افرادی که بلاک شده اند.

5.4 - Timeline

در این صفحه توییت‌های افرادی که کاربر فالو می‌کند و توییت‌های خودش نمایش داده می‌شوند. توییت‌های افراد میوت‌شده نمایش داده نمی‌شوند. فرد می‌تواند از طریق هر توییت با فردی که آن را نوشته یا خود توییت اینترکشن داشته باشد.

5.4.1 - ViewUser

کلاسی برای دیدن صفحه شخصی افراد که در صورت عدم بلاک بودن کاربر توسط آن فرد و پرایوت نبودن صفحه او، توییت‌های فرد برایش نمایش داده می‌شود. در این صفحه نیز در صورت وجود، گزینه‌های مربوطه نمایش داده می‌شوند و ماربر با دکمه back به راحتی می‌تواند به محل قبلی که از آن به صفحه این فرد آمده برگردد.

کاربرد این دکمه ساده است، یک حافظه موقت از جنس ArrayList دارد که آدرس محل قبلی که از آن به محل کنونی آمده‌ایم را در آن ذخیره می‌کند. این ArrayList با پیشروی به صفحات بیش‌تر طویل‌تر و با بازگشت به عقب کوتاه‌تر شده تا ما را به صفحه اصلی اولیه برساند.

5.4.2 - ViewTweet

همانند کلاس ViewUser ولی در این کلاس، هایsubTweet یک توییت (کامنت) و upperTweet آن (توییتی که توییت حاضر روی آن کامنت شده است) نیز نمایش داده می‌شوند و در صورت وجود دستورات آن‌ها نیز به نمایش در می‌آیند.

5.5 - Settings

بخش تنظیمات. در این بخش کاربر می‌تواند یوزرنیم، پسورد، تاریخ تولد، ایمیل، شماره تلفن و بیو خود را تغییر داده، دی‌اکتیو کند و تنظیمات شخصی پیج خود را تغییر دهد.

5.6 - Explore

این کلاس شامل دو بخش است. سرچ یوزر و سرچ توییت که هر دو به دنبال زیررشته‌ای درون یک رشته بلندتر می‌گردند. نتایج نیز توسط کلاس‌های SearchUsersCli و TweetsCli چاپ می‌شوند.

5.6.1 - RandomTweets

در این بخش توییت‌های رندومی از کاربران موجود در برنامه که کاربر ما آن‌ها را دنبال نمی‌کند، آن‌ها او را بلاک نکرده اند، او آن‌ها را میوت نکرده است و پیجشان پرایوت نیست نمایش داده می‌شوند. کاربر نیز می‌تواند با دکمه refresh این توییت‌ها را تغییر دهد.

5.7 - Notifications

این کلاس در صورتی که پیج کاربر پرایوت باشد دو بخش دارد. دیدن درخواست‌های فالو و دیدن نوتیفیکیشن‌ها. در بخش درخواست‌ها کاربر می‌تواند این درخواست‌ها را قبول یا رد کند.

در بخش نوتیفیکیشن‌ها نیز آن‌ها را به ترتیب زمانی خواهد دید. توضیحات کامل در بخش توضیح مدل User داده شده است.

5.8 - DirectMessages

در این کلاس فرد می‌تواند لیست چت‌های دایرکت خود را ببیند و در صورت لزوم با تایپ u/x به صفحه چتش با یک نفر برود.

5.8.1 - Groups

ساختار گروه در بخش مدل‌ها توضیح داده شد ولی در این بخش کاربر می‌تواند این گروه‌ها را ساخته و حذف و ویرایش نماید. هنگام share کردن توییت‌ها، با نوشتن عبارت g/x می‌توان این پیام را برای تمام افراد گروه x فرستاد.

5.8.2 - Chat

برای الگوریتم پیاده‌سازی بخش چت از مهرآفرین کاظمی مشورت گرفته شده است.

چت شخصی فرد با فرد دیگر. پیام‌های فرد و مخاطبش توسط رنگ‌های مجزا قابل تفکیک بوده و زمان هر پیام نیز مشخص است. برای رفرش صفحه تنها لازم است یک بار اسپیس زده شود و اینتر را بزنید.

می‌دونم بخاطر این کار قراره به جهنم برم، ولی، خروج از چت یک نفر مانند خروج از ادیتور Vim طراحی شده...
امیدوارم خدا من رو ببخشه...

5.8.3 - SavedTweets

یک بخش کاملا شبیه تایم‌لاین فقط با فرق این‌که توش کاربر می‌تونه کل پیامایی که سیو کرده رو ببینه.

فایل‌ها و پوشه‌ها

users

محل ذخیره‌سازی فایل‌های json کاربران. اسم هر فایل آی‌دی آن کاربر است.

tweets

محل ذخیره‌سازی فایل‌های json توییت‌ها. اسم هر فایل آی‌دی آن توییت است که شامل آی‌دی صاحب توییت و شماره توییت می‌باشد.

messages

محل ذخیره‌سازی فایل‌های json پیام‌های شخصی کاربران (دایرکت). اسم هر فایل آی‌دی آن پیام است.

database

محل ذخیره‌سازی اطلاعات موردنیاز برنامه برای ساخت یک اکانت، توییت یا مسیج

id.txt

در این فایل یک خط وجود داشته و آن نیز آخرین آی‌دی ایست که به یک کاربر اختصاص داده شده که در هنگام ساخت یک اکانت جدید، کاربر موردنظر آی‌دی ای متمایز از دیگر آی‌دی‌های موجود خواهد داشت.

emails.txt

در این فایل لیست تمام ایمیل‌های کاربران ذخیره می‌شود و در صورت تغییر ایمیل یک فرد ایمیل قبلی او با ایمیل جدیدش جایگزین می‌شود.

usernames.txt

مانند فایل emails.txt ولی برای یوزرنیم‌ها

phonenumbers.txt

مانند فایل emails.txt ولی برای شماره‌تلفن‌ها

message.txt

همانند فایل id.txt ، ولی این بار برای ذخیره سازی آخرین پیام ارسال شده.

log4j2.xml

فایل کانفیگ لاگر برنامه (که log۴j می‌باشد). برای ساخت لاگر و تنظیم آن از محمدامین رئیسی کمک گرفتم.

logs.log

محلی که اطلاعات لاگ در آن ذخیره می‌شود. برای موارد غیر مهم در آن از حالت info و debug ، برای موارد مهم از warn و fatal و برای موارد بسیار مهم از error استفاده شده است.