باسمه تعالی

دانشگاه صنعتی شریف

دانشکده علوم کامپیوتر

برنامه‌نویسی پیشرفته

فاز اول پروژه

دکتر بومری و استواری

طاها انتصاری

۹۵۱۰۱۱۱۷

در این گزارش سعی شده است که نکاتی چند در مورد نحوه پیاده‌سازی فاز اول پروژه برنامه‌نویسی پیشرفته ارائه شود. در این گزارش برای جلوگیری از شلوغی بیش‌از حد، به جای آوردن تمامی قسمت‌های موردنیاز در توضیح، به کد مربوطه ارجاع داده می‌شود.
با توجه به اینکه توجه این فاز از پروژه بر پیاده‌سازی بستر بازی است، مطابق توضیحات ارائه‌شده در فایل توضیح پروژه، قرار نیست که تمام امکانات بازی در این فاز پیاده‌سازی شوند و در این فاز تنها کلیات کارت‌های بازی و قهرمانان و ساختارهایی چند برای تغییر این ویژگی‌ها پیاده‌سازی شده‌اند.
در ادامه سعی شده است که قسمت‌های مختلف و نحوه‌ی پیاده‌سازی آن‌ها توضیح داده شوند. سعی شده است که ترتیب ارائه مطالب، همان ترتیب پیاده‌سازی آن‌ها در طول انجام این فاز باشد.

۱ Logging

برای پروژه یک لاگر اصلی در نظر گرفته شده است که وظیفه‌ی ذخیره‌ی اتفاقات برنامه، ذخیره اررها و دیگر اطلاعات مرتبط را دارد. مطابق درخواست پروژه نیز یک فایل لاگر دیگر برای هر بازیکن در نظر گرفته شده است که در آن کارهای کابر ذخیره می‌گردند.
برای پیاده‌سازی لاگر از کتابخانه java.util.logging.Logger استفاده شده است. برای ذخیره لاگرها هم از یک هندلر به فایل استفاده شد. نظر به تغییر رنگ‌بندی لاگر و بهترکردن فرمت خروجی آن بود که متاسفانه در اجرای آن به مشکلی خوردم که نتوانستم آن را حل کنم و در این فاز لاگرها در حالت سادی و دیفالت خود هستند.
کلاسی که لاگرها در آن نوشته شده‌اند در واقع یک singleton هست و لاگرهای یوزر و اصلی در آن به صورت استاتیک موجودند و در تمامی پروژه در هر جایی که لازم بوده است، اینستنس لاگر از این کلاس کپی گرفته شده است.

۲ Locations

برای سادگی پیاده‌سازی و هم‌چنین واقعی‌تر شدن پیاده‌سازی به یک بازی واقعی، مکان‌هایی به عنوان مکان‌های معتبر برای بازی در نظر گرفته شده‌اند که کاربر در هر آن می‌تواند تنها در یک مکان باشد. نکته‌ی دیگر این‌که کاربر نمی‌تواند در هر مرحله‌ای هر دستوری را اجرا کند. این مکانیزم علاوه بر این‌که یک سیستم واقعی است، ارر هندلینگ را ساده‌تر می‌کند.
کاربر در ابتدا در صفحه‌ی ورود بازی است و پس از ساخت حساب کاربری و یا وارد شدن به حساب خود به پنل کاربری برده می‌شود. تمامی قسمت‌های بازی از این پنل قابل دسترس هستند اما به عنوان مثال، دستور خرید کارت در این قسمت قابل اجرا نیست و بایستی ابتدا به قسمت فروشگاه رفت. نکته‌ی دیگر این‌که در صورتی که در فروشگاه هستیم، برای رفتن به قسمت کارت‌ها، ابتدا بایستی به پنل کاربری بازگشت.
این مکانیزم در ادامه و به خصوص در قسمت نشان دادن راهنمایی و فرمان‌های صحیح کمک به سزایی کرد.

Cards ۳

با توجه به این‌که در این فاز تنها مشخصات کارت‌ها لازم است و در واقع هیچ کدام از کارهای حمله، چارج [۱] و ... قرار نیست پیاده شوند، این قسمت به نسبت سرراست است. کلاس کلی card ویژگی‌هایی که بین همه انواع کارت‌های بازی مشترک هستند را ذخیره می‌کند و ۴ نوع کارت بازی، مینیون، جادو، سلاح و ماموریت [۲] از این کلاس به ارث می‌برند. با توجه به سادگی این قسمت در این پروژه، به جز کلاس مینیون و سلاح، دو کلاس دیگر فیلد جدیدی اضافه نمی‌کنند و تنها در سازنده خود، سازنده پدر را فرامی‌خوانند.
پس از این، با استفاده از سایت تعدادی کارت انتخاب شدند که این کارت‌ها در فایل InitiateCards تولید می‌شوند و با استفاده از روش ذخیره‌سازی توضیح داده‌شده در قسمت بعد، ذخیره می‌شوند تا در هر جایی از بازی که نیاز به کارت‌ها باشد، با بازسازی این فایل‌ها بتوان بدان‌ها دسترسی پیدا کرد. در فاز فعلی برای ساخت کارت‌ها نیاز به فراخوانی کلاس ذکر شده است که این کار در ابتدای هر بار اجرای CLI در صورت در خواست کاربر صورت می‌گیرد (با توجه به ضیق وقت نتوانستم این کار را اتوماتیک کنم که در صورتی که کارت‌ها موجود هستند دیگر آن‌ها را تولید نکند).

Serialization ۴

برای ذخیره سازی آبجکت‌های برنامه از اینترفیس جاوا تحت عنوان Serializable استفاده شده است. برای این‌کار تمامی آبجکت‌هایی که قرار است ذخیره شوند کافی‌است که این اینترفیس را implement کنند. نکاتی چند در مورد این اینترفیس:

- در صورتی که کلاس پدر آن را implement کند، کلاس‌های کودک نیازی به implement کردن آن ندارند.
- تمامی کلاس‌هایی که داخل یک کلاس استفاده شوند (composition) بایستی این اینترفیس را implement کنند.

این اینترفیس البته مشکلی دارد و آن هم این است که فایل خروجی آن در واقع بایت کد بوده و خوانا نیست. با توجه به این‌که در توضیح پروژه این مورد اشاره‌نشده بود که فایل ذخیره خوانا باشد و با راهنمایی استاد، تصمیم بر آن شد که از این روش ذخیره‌سازی استفاده شود. مزیت استفاده از این کار سادگی بیش از حد آن است.

[۱] charge
[۲] quest

Hero ۵

با توجه به نکات گفته‌شده، پیاده‌سازی کلاس هیرو نکته‌ی خاصی نداشته و کد آن به وضوح بیانگر منطق و رابط آن است.
برای هر هیرو ۱۰ کارت که یکی از آن‌ها از کلاس کارت‌های خودش بوده انتخاب شده و هنگام فراخوانی سازنده هیرو، اسم این ۱۰ کارت در آرایه‌ای ذخیره می‌شوند. دسته کارت‌های هر هیرو ۲۰ کارت جا داشته که هر کارت قادر به ۲ بار تکرار است. به جز ۱۰ کارت بالا، بقیه‌ی کارت‌هایی که هر هیرو قادر به دسترسی آن‌ها خواهد بود (مثلا از طریق خرید کارت) مجموعه‌ی تمامی کارت‌های خنثی باقی‌مانده و کارت‌های باقی‌مانده کلاس خود هیرو است.
مکانیزم خرید و فروش کارت و مکانیزم تغییر دسته کارت‌ها به همراه ارر هندلینگ‌های مرتبط پیاده‌سازی شده‌اند. نکته‌ی اضافی پیاده‌سازی شده، پیاده‌سازی ساختار پیشنهاد اسم کارت است که در قسمت بعدی توضیح داده‌شده است.

Closest Match ۶

در تمامی مراحلی که کاربر فرمانی را وارد می‌کند، در صورتی که فرمان مذکور در این مرحله موجود نباشد، از فرمان‌های مجاز این قسمت، هر یک که فاصله کمتری با فرمان وارد شده داشته باشند، به عنوان پیشنهاد به کاربر نشان داده می‌شوند تا کاربر در دستور بعدی بتواند فرم درست دستور را وارد کند.
به جز فرمان‌های سیستم، این الگوریتم بر روی اسم کارت‌ها و هیروها نیز پیاده‌سازی شده است و در صورت وقوع اشتباه، نزدیک‌ترین پاسخ ممکن به کاربر پیشنهاد داده می‌شود.
این سیستم با استفاده از فاصله‌ی Levenshtein [۳] پیاده‌سازی شده است. کد این برنامه از این سایت برداشته شده است.

User ۷

با پیاده‌سازی کلاس‌های کارت و هیرو، اکنون می‌توان کلاس کاربر را ساخت. مطابق توضیحات، هر کابر با سه هیرو ساخته می‌شود. نحوه ذخیره‌سازی کاربر نیز به کمک همان اینترفیس است. عموم توابع این کلاس، فراخوانی توابع کلاس هیرو هستند و بیشتر تنها در قالب یه واسط عمل می‌کنند.
رمز کاربر با استفاده از خانوانده SHA-۲۵۶ نگه‌داری می‌شود. چند ریزه‌کاری در مورد رمز کاربر نیز وجود دارد که مطابق اکثر سیستم‌های فعلی، رمز کاربر بایستی حداقل ۸ حرف بوده و حروف کوچک و بزرگ انگلیسی داشته باشد.

[۳] توضیح در سایت ویکی‌پدیا

CLI ۸

در این قسمت روند کلی یک اجرا از برنامه توضیح داده می‌شود. با اجرای برنامه ابتدا برنامه بررسی می‌کند که فولدر ذخیره‌سازی دیتا موجود بوده و در صورتی که موجود نباشد، فولدرهای ذخیره سازی را می‌سازد. در مرحله‌ی بعدی، در صورت درخواست کاربر آبجکت‌های کارت‌ها ساخته می‌شوند و در فولدر مخصوص خود ذخیره می‌شوند (این کار در اولین اجرای برنامه لازم بوده و در صورتی که به عنوان مثال ویژگی‌های کارت‌ها را تغییر دهیم نیز لازم‌الاجراس).
در هر مکانی و در وارد کردن هر دستوری دستورات exit، exit، help -a، back و Hearthstone help پشتیبانی می‌شوند. نکات مربوط به userID که در مورد اسم‌های کاربری حذف شده کاربرد دارد نیز پیاده سازی شده‌اند. برای وارد شدن به حساب کاربری، ابتدا رمز را با رمزی که در یک فایل که اطلاعات کلی تمام کاربران در آن ذخیره شده‌است، مقایسه کرده و در صورت صحت اطلاعات کاربر و آبجکت وی لود می‌شوند.
لازم به ذکر است که اسم دستورات دقیقا مشابه اسم‌های ذکر شده در فایل پروژه نیست و برخی تفاوت‌ها موجود است که البته با استفاده از help، این مورد مشکلی را به وجود نمی‌آورد.
در تمامی دستورات ارر هندلینگ‌های لازم و سیستم‌های هشدار مربوطه پیاده‌سازی شده است. یکی از مهم‌ترین قسمت‌ها برای این ارر هندلینگ، قسمت collection است که وظیفه تغییر دسته [۴] را بر عهده دارد. دلیل این نیز این بوده که دسته کارت هر هیروی بازیکن بایستی همواره تعداد ثابتی کارت (در فاز فعلی ۲۰ در نظر گرفته شده) داشته باشد. برای همین کار در این قسمت ابتدا تغییراتی که کاربر می‌دهند در یک آبجکت کپی ذخیره می‌شوند و تغییرات زمانی که تعداد کارت‌های دسته به ۲۰ برسد، به فایل اصلی منتقل می‌شوند (برای اضافه کردن کارت ابتدا بایستی کارتی را حذف کنیم که این کار تعداد کارت‌ها را به عنوان مثال به ۱۹ می‌رساند. در این حالت تغییرات ذخیره نشده‌اند و کاربر حتی قادر به خروجی از بازی نیست. بعد از اضافه کردن کارت به دسته کارت‌ها و رسیدن تعداد کارت‌ها به ۲۰، تغییرات اعمال می‌شوند.)

[۴] deck