



برنامه‌نویسی پیشرفته
زمستان 1402 - بهار 1403
فاز سوم پروژه

تعریف و طراحی پروژه:
آرین همتی



مقدمه‌ای بر فاز سوم (مثل همیشه حتماً بخوانید!)

به فاز سوم و پایانی پروره درس برنامه‌نویسی پیشرفته خوش آمدید. 🎉 شما یک بازی کامل شامل تعداد زیادی انتیتی با پروتکل‌های حرکتی متفاوت و کارکردهای پیاده‌سازی کردید و در طی این پروسه با فرایندهای پیشرفته گرافیکی، اصول معماری و طراحی و پارادایم‌های مختلف برنامه‌نویسی آشنا شدید و از آنها برای پیاده‌سازی یک کد توسعه‌پذیر و قابل فهم در راستای اصول **SOLID** استفاده کردید. همچنین شما در فاز دوم پروره توانستید با طراحی‌های **Event-Driven**، کارکردهای مربوط به فریم‌های مجزای بازی را به یکدیگر مرتبط و متصل کنید. در ادامه این فاز، با استفاده از اصول OOP، رفتارها و پروتکل‌های متفاوتی برای اجزای مختلف بازی را اعم از فریم‌ها و دشمن‌ها مختلط تعریف کرده و بدون ارتباط مستقیم آنها، رفتارهای متقابل بین این اجزا را کنترل کنید. به دنبال آن با مکانیزم‌های جدید **Global Routing**, **AOE** آشنا شدید و پروتکل‌های مختلفی برای نمایش فریم‌های بازی پیاده‌سازی کردید. در نهایت برای فرایندهای **ذخیره‌سازی بازی** و **مکانیک‌های Spawn** دشمن‌ها در صفحه بازی از روندهای **Serialization**, **Event Queuing** استفاده کردید و کار خود را در این فاز به پایان رسانید. از شما انتظار می‌رود در طول فازهای اخیر با استفاده از دانش خود در زمینه معماری و طراحی روندهای اجرا و Workflow، نرم‌افزاری هم‌خوان با معماری Event-Driven و بالاخص همسو با معماری MVC پیاده‌سازی کرده باشید که از توسعه‌پذیری مطلوبی برخوردار باشد. توصیه می‌شود با توجه به نسبت مهلت این فاز به محتوای مورد انتظار، در صورتی که کد شما در راستای این معماری‌ها پیاده‌سازی نشده است، برای پیاده‌سازی بدون اشکال این فاز زمان کافی برای اصلاح کد فازهای قبل خود اختصاص دهید. همچنین توجه کنید با توجه به ذات معماری‌های مبتنی بر شبکه، احتمال وقوع مشکلات **Concurrency** در روند اجرای برنامه بالا خواهد بود و توصیه می‌شود قبل از پیاده‌سازی این فاز، به رفع مشکلاتی از این قبیل هم توجه کافی داشته باشید. از آنجا که در این فاز به پروسه‌های پیشرفته‌تر ذخیره‌سازی احتیاج خواهد داشت، ممکن است برای پیاده‌سازی فرایندهای **Serialization Annotation**, **Deserialization** به Reflection در یک از سری‌های تمرين درس (در موضوع Reflection) از برنامه درسی حذف شده و نمره آن به این فاز انتقال پیدا کرده است. لذا از آنجا که در طراحی بازی‌ها، **Responsive** بودن همواره یکی از مهم‌ترین اصول پیاده‌سازی است، خوب است برای جلوگیری از مشکلات احتمالی استفاده از Reflection در یک کد جاوا، با تغییراتی که این کار لحاظ زمانی و منابع برای اجرای بازی شما ایجاد می‌کند آشنا باشید. [این لینک](#) برای مطالعه درباره این دست از رفتارهایی که به علت استفاده از Reflection ایجاد می‌شوند منبع مفیدی خواهد بود. اهداف آموزشی این فاز شامل مباحثی اعم از Network, Data Stream, Reflection از دانشی که در طول درس درباره این مباحث کسب کرده‌اید، در قسمت‌های مختلف کد خود از پروتکل‌های ارتباطی صحیح و مناسب استفاده کنید و معماری‌های Stream متفاوتی برای رد و بدل کردن Request/Response های خود به کار بگیرید.

نکته بسیار مهم: در تعریف فاز آخر پروره، با توجه به مهلت کمتر آن نسبت به فازهای قبل عمده‌تر سعی در این راستا بوده که محتوای فاز، غالباً حول محور اهداف آموزشی باشند و از تعریف آیتم‌های جدید در این فاز جلوگیری شود. لذا بدیهی است که بخش قابل توجهی از نمره این فاز، به صورت مستقیم یا غیرمستقیم وابسته به پیاده‌سازی آیتم‌های فازهای قبل است. با اینکه سعی بر این است که این وابستگی کمینه باشد، در حدود یک نمره از 3.5 نمره این فاز (با احتساب نمره اضافه شده از تمرين Reflection) مربوط به آیتم‌های Reflection) می‌شود. در صورتی که فاز اول و دوم را به طور کامل پیاده‌سازی نکرده‌اید، که باقی نمره این فاز با استفاده از ابزارهای فاز اول پروره قابل پیاده‌سازی باشد. در صورتی که قرار داده شده است برای فاز سوم پروره استفاده کنید. ویدیویی برای توضیح داده خواهد شد) با استفاده می‌توانید از کدی که در [این لینک](#) برای شما قرار داده شده است برای فاز سوم پروره استفاده کنید. ویدیویی برای توضیح داده خواهد شد) با استفاده از لینک ضمیمه شده است. دقت کنید در صورت استفاده از هر قسمی از کد قرار داده شده، از شما انتظار می‌رود مانند باقی قسمت‌های کد ارسالی‌تان به تمام جزئیات این بخش‌ها نیز مسلط باشید و در طی فرایند تحویل و ارزیابی، قادر به ایجاد تغییرات در این کد باشید. در حدود 20 درصد نمره فاز نیز به موضوعات مربوط به کلین‌کدینگ و رعایت صحیح و بجای معماری‌ها و دیزاین‌پترن‌های مختلف اختصاص خواهد داشت.



2.....	مقدمه‌ای بر فاز سوم (مثل همیشه حتماً بخوانید!)
3.....	این همه چیز رو عوض نمیکنه!.....
4.....	بازی آنلاین!
4.....	Leaderboard
4.....	قابلیت بازی آفلاین!
4.....	Data Integrity Validation (DIV)
5..... Squad
5..... Squad Battle
5..... Squad Vault
6..... Battle Modes
6..... Monomachia (نبرد تن به تن)
6..... Colosseum (نبرد دونفره)
6..... Battle Summon
7..... Squad Battle پایان
7..... Server دیتا هندلینگ سمت
7..... Wave ها خودکارسازی حملات
8..... اخیرین سری بخش‌های امتیازی!
8..... کنسول سرور
8..... Database
8..... Global Exception Handling

این همه چیزو عوض نمیکنه!

در صحنه **پایانی فاز قبل** Smileی از بین رفته و فریم بازی از کنترل او خارج می‌شود. او که در فکر نجات خود از مخصوصه است، با استفاده از ابزارهای Reflection متوجه می‌شود اساساً هیچوقت Encapsulation ای که **شما را از دسترسی او ایمن نگه داشته**، مانعی برای دسترسی او به کارکتر شما ایجاد نمی‌کند و او می‌تواند به راحتی به اطلاعات شما دسترسی پیدا کند. او که دیگر توان مقابله با **Epsilon** را ندارد، تصمیم می‌گیرد قوی‌ترین سلاحی که در اختیار دارد را برای مقابله با شما به کار بگیرد: **Epsilon**. او سپس تمام فیلدهای **Epsilon** را با استفاده از دور زدن فیلدهای Private با متدهای Reflection کپی کرده و برای خود **Epsilon** دیگری می‌سازد تا شما را در مبارزه شکست دهد. آیا می‌توانید بر قدرت‌های خود غلبه کنید؟

بازی آنلاین!

یک از اهداف اصلی این فاز، تبدیل بازی‌ای که در فازهای قبلی پیاده کرده‌اید به یک بازی شبکه محور است. در این فاز شما باید کد خود را در قالب دو برنامه مجزای جاوا تحویل دهید که هر یک به طور مجزا مسئولیت کارکردهای Server, Client را بر عهده می‌گیرند. اگر در فازهای قبلی به درستی از معماری MVC بهره گرفته باشید، این گام برای شما بسیار ساده خواهد بود. وقت کنید از آنجا که هر یک از برنامه‌های مربوط به کدهای ارسالی Server باید مستقل‌باشد و به درستی قابل اجرا باشند، هیچ‌گونه Dependency مستقیمی بین این دو وجود داشته باشد. اما بدیهیست برای فعالیت همزمان و Synchronized بین این دو قسمت، ارتباطات بین Client, Server مورد نیاز است تا:

1. فعالیت ورودی کاربر را از سمت Client به سمت Server مخابره کرده و باعث پیش‌روی بازی شوند
2. در هر فریم اطلاعات جدید بازی را (پس از پیش‌روی) از سمت Server (جهت نمایش فریم جدید) به Client مخابره کنند

به این منظور از شما انتظار می‌رود در قسمت Server, Client کد ارسالی خود وظایف زیر را پیاده‌سازی کنید:

1. در شروع اجرای بازی از سمت Client، تلاش کنید به آدرس از پیش تعیین شده‌ای مربوط به Server متصل شوید
2. در صورت موفقیت در اتصال به این آدرس، پیامی به کاربر نمایش دهید و منوی اصلی بازی را (مانند شروع فاز دوم) نمایش دهید
3. در صورت عدم موفقیت در اتصال به آدرس Server، این مشکل را به همراه گزینه‌ای برای تلاش مجدد به کاربر اطلاع دهید. همچنین باید گزینه‌ای برای اجرای بازی بدون اتصال به Server در نظر بگیرید. (توضیحات لازم در ادامه آمده است) در هر صورت (اجرای بازی با اتصال به Server یا بدون آن) شما باید گزینه‌ای برای نمایش وضعیت اتصال و قطع/وصل کردن ارتباط با Server در منوی اصلی بازی در نظر بگیرید. همچنین در صورت قطع شدن ارتباط با Server در هر نقطه‌ای از بازی، باید آن را در قالب یک پیام به کاربر اطلاع دهید
4. قسمت Server کد ارسالی شما باید همواره آماده دریافت اتصال از سمت یک Client جدید بوده و لیست کاملی از Client های متصل به Server داشته باشد. همچنین Server شما باید همواره آماده دریافت Request های hrmزن باشد و پاسخ دادن به آنها باشد.

به منظور انجام این ارتباطات از شما انتظار می‌رود در هر مورد از پروتکل ارتباطی مناسب (QUIC, UDP, TCP,...) برای انتقال داده‌های موردنیاز استفاده کنید. همچنین استفاده صحیح از دیزاین‌پترن‌های ارتباطی (Request-Response, Pulling,...) موردنانتظار است. برای مطالعه بیشتر درباره انواع دیزاین‌پترن‌های ارتباطی می‌توانید [این منبع](#) را مطالعه کنید. همچنین مطالعه [این لینک](#) برای یادگیری بهتر QUIC در سودمند خواهد بود. **توجه کنید که Server شما باید بتواند پاسخگوی چند Client همزمان باشد و پیام‌های دریافتی/ارسالی را بدون تداخل دریافت و پردازش کند**

Leaderboard

در این بخش از شما انتظار می‌رود در پایان هر بازی یک جدول امتیازات کامل برای کاربر نمایش دهید که در آن زمان زنده ماندن و XP به دست آمده او در آن بازی به همراه رکوردهای بیشترین زمان زنده ماندن و بیشترین XP کسب شده در یک بازی در بین همه بازیکنان دیگر (با احتساب خودش) با نام کاربری آنها نمایش داده شده است. همچنین می‌توانید آمارهایی از قبیل تعداد دفعات مرگ کاربر در بازی را به او نمایش دهید.

قابلیت بازی آفلاین!

با اینکه هدف این فاز از پروژه پیاده‌سازی بازی به صورت آنلاین است، از شما انتظار می‌رود بتوانید بازی را طوری طراحی کنید که کاربر کمکان قادر به بازی آفلاین و بدون اتصال به Server باشد. به این منظور شما باید نتایج بازی‌های آفلاین کاربر را برای ارسال به Server در اولین فرصت، در فایل ذخیره کنید. همچنین تبعاً قابلیت‌های آنلاین بازی در این حالت برای کاربر غیر فعال خواهد بود.

Data Integrity Validation (DIV)

یکی از مشکلات همیشگی پیاده‌سازی بازی‌های Multiplayer آنلاین، اطمینان حاصل کردن از درستی داده‌های غیر مطمئن است. برای مثال یک کاربر صرفاً با کشف Response های ارسالی به سرور پس از پایان بازی می‌تواند امتیاز دلخواه و غیرواقعی از نتیجه بازی‌های آفلاین خود را به سمت Server مخابره کند. برای جلوگیری از این مشکلات، شما باید متناسب با معماری و نحوه پیاده‌سازی خود روش‌هایی برای اطمینان حاصل کردن از دیتای ارسالی به سمت Server طراحی و اجرا کنید. برای آشنایی با یکی از راه حل‌های معمول برای این مشکل می‌توانید [این لینک](#) را مطالعه کنید.

Squad های بازی

شما در هر Server باید لیستی از تعدادی Squad حفظ و آپدیت کنید. کاربری Squad ها در ادامه داک توضیح داده خواهند شد. شما باید دکمه‌ای برای عملیات‌های مربوط به Squad در منوی اصلی بازی در نظر بگیرید تا هر یک از کاربران بتوانند با انتخاب این گزینه، وارد این بخش شوند:

1. در ابتدای ورود به این بخش کاربر باید یک نام کاربری برای خود انتخاب کند. هر کاربر در حداقل یک Squad می‌تواند حضور داشته باشد.
2. در این بخش هر کاربر بدون Squad باید بتواند لیست کاملی از تمام Squad های Server را با تعداد کاربران حاضر در آنها مشاهده کند و درخواست عضویت در آنها بدهد. این درخواست از طریق یک پیام به اطلاع مالک Squad می‌رسد و سپس عضویت قبول یا رد می‌شود و پاسخ آن مجدداً به اطلاع کاربر خواهد رسید. طبیعتاً قبول کردن عضویت بازیکنی که عضو Squad دیگری شده است امکان‌پذیر نیست.
3. هر کاربر بدون Squad می‌تواند با پرداخت 100 واحد XP، یک Squad جدید تاسیس کند و به عنوان مالک Squad شناخته شود و بنابراین می‌تواند Squad را حذف کند. در صورتی که یک Squad حذف شود، اعضای آن به طور خودکار از آن بیرون خواهند رفت. مالک Squad می‌تواند با انتخاب هر عضو، او را از Squad حذف کند. اگر تعداد اعضای Squad به صفر برسد، به طور خودکار حذف خواهد شد.
4. هر کاربری که در یک Squad حضور دارد باید بتواند با انتخاب یک گزینه، از Squad فعلی خود خارج شود. او همچنین باید بتواند در این بخش لیست کاملی از نام و XP و وضعیت اعضای Squad خود مشاهده کند. (وضعیت یک کاربر در ادامه توضیح داده شده است)

Squad Battle

شما باید یک سیستم مسابقه و رده‌بندی بین Squad های بازی طراحی کنید. در این روند، شما باید متدهای `initiateSquadBattle()` را در سمت Server کد خود پیاده‌سازی کنید که با فراخوانی آن از طریق CLI، برنامه شما باید Squad ها را به صورت تصادفی به جفت‌های دوستی تقسیم کند. حال اگر دو Squad مختلف برای نبرد با یکدیگر انتخاب شده باشند، این موضوع از طریق یک پیام به هر یک از اعضای این دو اطلاع داده خواهد شد. (می‌توانید برای نمایش این پیام از کلاس JOptionPane استفاده کنید) این پیام در اولین زمانی که کاربر به Server متصل باشد و در منوی اصلی بازی قرار داشته باشد، به اطلاع او خواهد رسید. شما باید تمام قابلیت‌های زیر را در این بخش پیاده‌سازی کنید: (همچنین شما می‌توانید در این قسمت بخش امتیازی مربوطه را پیاده‌سازی کنید)

1. در هر Squad Battle، اعضای هر یک از Squad ها باید بتوانند در قسمت مربوط به Squad در منوی اصلی خود، اعضای تیم حریف را با نام کاربری، XP و وضعیت آنها را مشاهده کند. وضعیت هر بازیکن شامل Online, Offline, Busy می‌باشد و وضعیت به معنای در حال بازی بودن بازیکن خواهد بود. **توجه کنید این اطلاعات باید در حین مشاهده، آپدیت شوند و ثابت نخواهند بود.**
2. هر کاربر یک Squad باید بتواند اطلاعات مربوط به Squad Battle های قبلی و فعلی خود را (از زمان عضویت) در قسمت Squad منوی اصلی مشاهده کند. این اطلاعات می‌تواند در قالب یک لیست از نتایج به کاربر نمایش داده شود.
3. در طول مدت Squad Battle، بازیکنان هر دو Squad باید تا حد ممکن از طریق Colosseum و Monomachia Battle Mode های Squad Battle را مشاهده کنند. (که در ادامه داک توضیح داده خواهند شد) در انتهای Squad Battle به هر یک از اعضای Squad برنده 500 واحد امتیاز کسب کنند. (در ادامه داک توضیح داده خواهند شد) در انتهای Squad Battle به هر یک از اعضای Squad بازنده 300 واحد XP کسر خواهد شد. (طبیعتاً این مقدار از صفر کمتر نخواهد شد)
4. در طی یک Squad Battle هر بازیکن تنها یک بار می‌تواند در هر یک از دو Battle Mode مذکور بازی کرده و امتیاز کسب کند. در هر Squad Battle تنها کاربرهای قابلیت نبرد در خواهند داشت که پیش از شروع Squad Battle وارد Squad Battle شده باشند.

Squad Vault

در این بخش از قسمت Squad در منوی اصلی، بازیکن باید خزانه Squad را مشاهده کند. هر بازیکن در طول هر Squad Battle می‌تواند مقداری انتخابی تا سقف 200 واحد XP به خزانه پرداخت کند. مالک Squad می‌تواند با استفاده از XP های خزانه Squad قابلیت‌های زیر را فعال کند:

- **Call of Palioxis**: با پرداخت (Number of squad members in battle × 100) واحد XP از خزانه Squad، میزان کسر XP از هر بازیکن (در صورت باخت در Squad Battle) را به 100 کاهش دهد. در صورت بردن Squad، این قابلیت برای Battle بعدی استفاده خواهد شد.
- **Call of Adonis**: با پرداخت 400 واحد XP از خزانه Squad، قابلیت یک بار استفاده از Summon را در نبردهای Monomachia برای تمام Squad فعال کند. توضیحات کامل درباره نحوه عملکرد این قابلیت و نحوه استفاده از آن در بخش بعدی داک آمده است.
- **Call of Gefjon**: با پرداخت 300 واحد XP از خزانه Squad، جایزه برد و جریمه باخت Squad خود در Battle جاری را دو برابر کند.

Battle Modes

Monomachia (نبرد تن به تن)

برای شروع این نوع نبرد، کاربر باید از لیست بازیکنان Squad حرفه، یک بازیکن با وضعیت Online که از ابتدای این Squad Battle در هیچ نبرد Monomachia ای شرکت نداشته را انتخاب کند. (در صورت نداشتن هر یک از این شرایط باید هشدار یا پیام مناسب به کاربر نشان داده شود) در ادامه درخواستی برای نبرد Monomachia برای بازیکن حرفه (در قالب یک پیام) ارسال می‌شود و بازیکن حرفه می‌تواند این درخواست را تایید یا رد کند. نتیجه این درخواست در قالب یک پیام به اطلاع کاربر خواهد رسید. در صورت تایید این درخواست، هر دو بازیکن 15 ثانیه زمان خواهند داشت تا آمادگی لازم برای شروع نبرد را انجام دهند. (می‌توانید زمان باقی‌مانده را در گوشه صفحه نمایش دهید) پس از پایان زمان، بازی آغاز می‌شود و دو بازیکن در قالب دو کاراکتر Epsilon با رنگ‌های مختلف در صفحه بازی ظاهر می‌شوند. در این حالت نبرد، دشمن‌هایی که دائمًا به سمت حرکت می‌کنند (اعم از Trigorath, Quarantine) و یا به طور مستقیم Epsilon را هدف قرار می‌دهند (مثل Omenoct) به صورت جفتی در صفحه بازی Spawn می‌شوند و هر یک از این دو دشمن، یکی از این دو کاراکتر Epsilon را مورد هدف قرار می‌دهد. تیرهای هر یک از کاراکترهای Epsilon به کاراکتر دیگر آسیب می‌زند و بازیکنی که زودتر بمیرد، برنده بازی خواهد بود. پس از 10 دقیقه در صورتی که هیچ یک از بازیکن‌ها نمرده باشند، بازیکنی که HP کمتری دارد بازده است و در صورت داشتن HP برابر بازی با تساوی به پایان می‌رسد. در این حالت بازی، باس ظاهر نمی‌شود و صرفاً Wave های مختلف از دشمن‌های دیگر سد راه بازیکنان خواهند شد. بازیکن برنده 80 واحد XP به دست خواهد آورد.

Colosseum (نبرد دونفره)

برای شروع این نبرد، کاربر باید از لیست بازیکنان Squad خودش، یک بازیکن با وضعیت Online که از ابتدای این Squad Battle در هیچ نبرد Colosseum ای شرکت نداشته را انتخاب کند. (در صورت نداشتن هر یک از این شرایط باید هشدار یا پیام مناسب به کاربر نشان داده شود) در ادامه درخواستی برای این نوع نبرد برای هم‌تیمی‌اش (در قالب یک پیام) ارسال می‌شود و بازیکن می‌تواند این درخواست را تایید یا رد کند. نتیجه این درخواست در قالب یک پیام به اطلاع کاربر اولیه خواهد رسید. در صورت تایید این درخواست، هر دو بازیکن 15 ثانیه زمان خواهند داشت تا آمادگی لازم برای شروع نبرد را انجام دهند. (می‌توانید زمان باقی‌مانده را در گوشه صفحه نمایش دهید) پس از پایان زمان، بازی آغاز می‌شود و دو بازیکن در قالب دو کاراکتر Epsilon با رنگ‌های مختلف در صفحه بازی ظاهر می‌شوند. در این حالت نبرد، دشمن‌ها با سرعت و قدرت بالاتری Spawn شده و به صورت تصادفی به سمت یکی از دو کاراکتر حرکت می‌کنند. در این حالت از بازی تیرهای دو کاراکتر Epsilon به یکدیگر آسیب نخواهند زد. در صورت شکست دادن باس آخر بازی (Smiley) پیش از مرگ هر دو بازیکن، هر یک از بازیکنان 30 واحد XP به دست خواهد آورد.

Battle Summon

در این قابلیت شما می‌توانید در صورت داشتن حداقل یک Summon استفاده نشده در Squad خود، یکی از اعضای Online در Squad خود را در نقش **بازیکن فرعی** برای کمک در یک نبرد Monomachia فرا بخوانید. در صورتی که بازیکن Online، درخواست کاربر را قبول کند، یکی از های قابل استفاده Squad کسر می‌شود و این بازیکن برای کمک به کاربر وارد بازی می‌شود. در این صورت این بازیکن نقش مشابه هم‌تیمی بازی در حالت Colosseum ایفا خواهد کرد. (دشمن‌های بازی به طور تصادفی به یکی از این دو بازیکن حمله می‌کنند و تیرهای این دو بازیکن بر روی یکدیگر اثر نخواهد داشت) به علاوه در صورتی که بازیکن اصلی در بازی کشته شود، بازیکن Summon همچنان به نبرد با حرفه ادامه خواهد داد.

- در صورتی که هر یک از بازیکنان اصلی یک نبرد Colosseum به علت قطع ارتباط با Server از بازی خارج شود، بازیکن دیگر تا پایان به بازی ادامه خواهد داد. مشابهًا در صورت حضور هم‌زمان بازیکن اصلی و Summon او به عنوان بازیکن فرعی در یک بازی Monomachia و قطع ارتباط هر یک از این دو بازیکن با سرور و خروج او از بازی، بازیکن دیگر تا زمان مرگ و یا کشته شدن حرفه به بازی ادامه می‌دهد.
- در صورتی که زمان رویداد Squad Battle در طی بازی، به Server پرسد، بازی نیمه‌کاره متوقف شده و هر Squad صرفاً به اندازه XP جمع‌آوری شده در طی بازی، به XP خود اضافه خواهد کرد. (بنابراین این مقدار در نبرد Colosseum دو برابر نخواهد شد)
- از آنجا که تعدادی از قابلیت‌های مهم بازی در منوی توقف بازی قرار دارد، امکان توقف بازی در هر یک از دو Battle Mode معرفی شده وجود دارد و در صورت توقف هر یک از بازیکنان، بازی برای تمام بازیکنان متوقف خواهد شد اما زمان گذرانده شده هر بازیکن در منوی توقف بازی محدود به 45 ثانیه است و در صورت استفاده از تمام زمان توقف توسط هر بازیکن، دیگر امکان توقف بازی برای او وجود نخواهد داشت. همچنین توجه کنید مدت زمان گذرانده شده در حالت توقف از بازیکنی که بازی را متوقف کرده کسر خواهد شد.
- تحت هر شرایطی، در هر نبرد Monomachia تنها یک بار استفاده از قابلیت Summon توسط هر یک از Squad ها امکان‌پذیر است.

دقت کنید در حین Squad Battle امکان پرداخت به خزانه و خرید قابلیت برای Squad وجود ندارد و این موارد فقط قبل شروع Battle قابل انجام هستند

پایان Squad Battle

در نهایت شما باید بتوانید از طریق اجرای متدهای `SquadBattle` که ارسالی شما پیاده‌سازی شده باشد (terminateSquadBattle) در سمت Server که باید در سمت Squad (که باید در سمت Server کد ارسالی شما پیاده‌سازی شده باشد) جمع آوری شده برای Squad های مختلف را به پایان برسانید. در آخر XP جمع آوری شده برای Squad های رقیب مقایسه می‌شوند:

1. هر Squad که Squad Mode تعريف شده کسب کرده باشد، برنده Battle خواهد بود.
2. در صورت تساوی در XP کسب شده، Squad ای که بردهای بیشتری در نبردهای Monomachia کسب کرده باشد برنده خواهد بود.
3. در صورت تساوی در این مقادیر، Squad ای که قابلیت **Call of Gefjon** را فعال کرده باشد پیروز Battle خواهد بود.
4. در نهایت در صورتی که هر دو Squad در تمام موارد قبل برابر باشند، برنده به صورت تصادفی انتخاب خواهد شد!

دیتا هندلینگ سمت Server

در این فاز، همانطور که منطقی و واضح بنظر می‌رسد، شما جز در حالت بازی آفلاین، تمام داده‌های همه بازیکنان را در سمت Server کد خود ذخیره می‌کنید و سپس با استفاده از روندهای اجرای مختلف (اعم از On-call, Consistent) دیتابیس مورد نیاز برای اجرای همروند دو بخش مجزای بازی خود را از Server دریافت و یا به آن ارسال می‌کنید. از این رو شما نیازمندی تمام اطلاعات مربوط به هر کاربر شامل نام کاربری، Squad مربوطه، مقدار XP، قابلیت‌های باز شده در بازی و قابلیت فعال، تاریخچه امتیازات و غیره را در قالب یک فایل به ازای هر کاربر ذخیره و دائمًا آپدیت کنید. برای تشخیص فایل سیو مربوط به هر کاربر می‌توانید Mac Address سیستم کاربر را در بدو اتصال به سرور فراخوانی کرده و فرایند شناسایی کاربر در روند ذخیره‌سازی اطلاعات را با استفاده از این مقدار انجام دهید. همچنین اطلاعات مربوط به بازی شامل تمام مقادیر ثابت مطلق، (مقادیر ثابتی که از ثوابت دیگر بازی محاسبه نمی‌شوند و ذاتاً ثابت هستند) اطلاعات مربوط به Squad ها اعم از XP جمع آوری شده و اعضا و نبردهای بین Squad ها و تمام موارد دیگر که نیاز به ذخیره‌سازی دارند را مطابق **روند ذخیره‌سازی در فاز اخیر** در قالب چند فایل ذخیره کنید. پیاده‌سازی **موارد امتیازی مربوطه فاز دوم** در این فاز توصیه می‌شود. در این قسمت می‌توانید قسمت امتیازی این بخش را نیز پیاده‌سازی کنید.

خودکارسازی حملات Wave ها

در این فاز از شما انتظار می‌رود روند پیاده‌سازی Wave های مختلف دشمن را به صورت خودکار پیاده‌سازی کنید و از فراخوانی دستی دشمن‌ها در مناطق مشخص بپرهیزید. در گام اول از شما انتظار می‌رود با داشتن لیستی از دشمن‌های موجود در بازی و حداقل Wave برای ظاهر شدن آنها، (برای مثال Smiley به وضوح نباید در Wave اول Spawn شود) صرفاً با دریافت تعداد Wave های موجود در بازی را به طور خودکار سلسه مراتبی از Wave ها ایجاد کنید. نمونه‌ای از این فرایند خودکار در کلاس WaveManager کد **فاز اول که در اختیار شما قرار گرفته** پیاده‌سازی شده است. این کار به منحصر به فرد بودن تجربه بازیکن حتی بعد از چندین بار پایان بازی کمک کرده و کاربر را مجبور به استفاده از استراتژی‌های متفاوت می‌کند. در گام دوم از شما انتظار می‌رود با استفاده از ابزارهای Reflection، به طور خودکار دشمن‌های موجود در بازی را به عنوان Subclass های یک کلاس اصلی دشمن (که طبق اصول OOP باید کلاس تمام دشمن‌های دیگر خود را از آن به ارت برده باشید) شناسایی کرده و فرایند ایجاد و اجرای Wave های حمله در بازی را تقریباً به طور کامل خودکار کنید. با این کار، در صورتی که در طی توسعه بازی، دشمن‌های جدیدی به بازی اضافه شود، نیازی به اضافه کردن دستی آنها در فرایند اجرای Wave های مختلف بازی نخواهد بود که اصطلاحاً هزینه توسعه برنامه را کاهش می‌دهد. همانطور که در مقدمه فاز اشاره شد، از شما انتظار می‌رود با تسلط به روندهای اجرا در متدهای پکیج Reflection، از ایرادهای Workflow ناشی از استفاده از متدهای این پکیج در Coupling Point های کد خود جلوگیری کنید و لذا رعایت صحیح این موارد در نمایش این بخش موثر خواهد بود.

یادداشت پایانی!

واقعاً خسته نیاشید! امیدوارم این ترم و این پیروزه برآتون خوب و آموزنده بوده باشه و درین این همه مطالب زدوم و عجیب غریب به چیزهای علاقمند شده باشید که باعث بشه بردند ندانشون رو بگیرید و شاید دربارشون بیشتر مطالعه کنید! من و تیم درس برآتون عمیقاً آرزوی موفقیت و سریاندی می‌کنیم و امیدواریم با هر نمره‌ای هم که این درس رو به پایان رسوندید، از یادگیریش لذت برده باشید و این درس برآتون یه تجربه جالب (البته با وجود پر زحمت بودن ) بوده باشه. امیدواریم سال‌های بعد هم شما توی تیم درس باشید و یه ورودی دیگه رو با چیزای جالب درباره برنامه‌نویسی آشنا کنید! برای همتون آرزوی موفقیت من کنیم و امیدواریم همیشه از یاد گرفتن لذت ببرید. با آرزوی موفقیت، تیم درس برنامه‌نویسی پیش‌رفته...

آخرین سری بخش‌های امتیازی!

کنسول سرور

در این بخش شما باید برای اجرای دستورهای تعریف شده در Server در طول Runtime، به کمک لایبریری‌هایی اعم از Picocli، برنامه‌ای با قابلیت Command Line در کد سرور خود پیاده‌سازی کنید تا دستوراتی مانند `initiateSquadBattle`, `terminateSquadBattle` در حین اجرای برنامه قابل اعمال باشند. همچنین می‌توانید برای تغییر سرعت بازی، تغییر مقادیر ثابت اعم از سرعت و شتاب دشمن‌ها و Epsilon دستوراتی تعییه کنید. برای مطالعه بیشتر درباره این لایبریری و نحوه کار با آن می‌توانید به [این لینک](#) مراجعه کنید.

Database

یکی از ابزارهای مناسب برای ذخیره‌سازی داده‌های دسته‌بندی شده، استفاده از پایگاه‌های داده یا Database است. برای آشنایی مقدماتی با دیتابیس‌ها مطالعه [این منبع](#) سودمند خواهد بود. با اینکه ارتباط مستقیم مدل‌های در حال اجرا در Runtime با پایگاه داده از طریق JDBC, JPA و ابزارهای مشابه امکان‌پذیر است، استفاده از ابزارهای ORM به علت هم‌سویی با پارادایم‌های برنامه‌نویسی OOP و معماری MVC توصیه می‌شود. برای مطالعه درباره ORM‌ها می‌توانید [این لینک](#) را دنبال کنید. برای استفاده از ORM در جاوا Hibernate ORM شاید یکی از بهترین گزینه‌ها باشد. برای این بخش موارد زیر مورد انتظار خواهد بود:

1. استفاده صحیح و بجا از جدول‌ها و تایپ درست برای ستون‌های دیتابیس
2. تعریف روابط صحیح بین جدول‌های استفاده شده در ذخیره‌سازی داده‌های بازی
3. ذخیره‌سازی کامل داده‌ها در دیتابیس و عدم ذخیره‌سازی دیتا در قالب فایل‌های JSON
4. تخصیص صحیح ستون‌های مربوط به Primary Key, Index
5. در صورت عدم اتصال به Server، داده‌ها به صورت Local ذخیره شده و انتقال به Server در اولین فرصت بعد از اتصال

Global Exception Handling

یکی از قابلیت‌های کارای فریمورک Spring Boot نسبت به جاوا، تفاوت در نحوه Exception Handling آن است. همانطور که می‌دانید، فرایند Exception Handling در جاوا عمده‌تاً با بلوک‌های try-catch انجام می‌گیرد. این در حالی است که در Spring Boot فرایند `@ControllerAdvice` قابلیتی به نام Global Exception Handling انجام می‌شود. در این رویه با استفاده از `Annotate` کردن یک کلاس با `Annotation` تمام Exception‌هایی که در هر Controller اتفاق می‌افتد را به این کلاس Redirect می‌کنیم. در ادامه در این کلاس متدهایی با `@ExceptionHandler` ایجاد می‌کنیم که رویه‌های هندل کردن Exception شده به این کلاس را در بر خواهند داشت. در اینجا شما می‌توانید با `throw` کردن انواع `ExceptionMessage` ها و Java Error ها، خروجی قابل قبولی را در صورت وقوع Exception به کاربر نمایش دهید. مرکزیت بخشیدن به فرایند `Exception Handling` باعث می‌شود هزینه توسعه‌پذیری کد شما را از لحاظ هندل خودکار Exception کاهش می‌دهد و باعث می‌شود فرایندهای `Exception Handling` در کدتان پایدار و قابل پیش‌بینی باشند. از طرف دیگر این عملیات فرایند Logging را در سطوح Warning, Severe Logging بهبود می‌بخشد و Log کردن این سطح از کد را بسیار ساده و مجتمع می‌کند. وظیفه شما در این قسمت این است که با استفاده از ابزارهای Reflection در جاوا، رویه‌ای مشابه Global Exception Handling در Spring Boot را در جاوا پیاده‌سازی کنید. پیاده‌سازی Annotation‌های مربوطه در این قسمت ضروری است، لذا توصیه می‌شود پیش از پیاده‌سازی آن لینک‌های زیر را مطالعه کنید:

1. [Java Annotations \(Oracle Documentation\)](#)
2. [Java Errors, Java Exceptions \(Oracle Documentation\)](#)
3. [Lesson on Java Exception Handling \(Oracle Documentation\)](#)
4. [Lesson on Spring Boot Global Exception Handling](#)

با آرزوی موفقیت روزافزون
تیم درس برنامه‌نویسی پیشرفته
1403.04.25

