Manager

نام تابع	خط	توضیحات
Boolean moveIsValid(int row, char col)	63	سطر و ستون را ورودی می گیرد درصورتی که مجاز به حرکت در آن خانه باشید ترو وگرنه فالس برمیگرداند
String kingPosition(Color color)	34	رنگ را ورودی میگیرد و بین مهره ها، سطر و ستون شاه آن رنگ را پیدا میکند و بعنوان یک استینگ که کرکتر اول آن ستون و ادامه آن شماره ی سطر است برمیگرداند
Void check4check(Color color)	67	در زمانی که این تابع در حال اجراست، بولین check2Calledمقدار ترو دارد* رنگ کیش دهنده را ورودی می گیرد و برای تمام مهره های آن رنگ بجز شاهش(شاه کیش نمیدهد)، تابع()possibleMoves را با چک2کالدِ ترو اجرا میکند. حالا با استفاده از()kingPosition و ساب استرینگ و گر ات و استرینگ ولیو آو، استرینگ را تفکیک و سطرو ستون را بدست میاورد. اگر رنگ زمینه ی خانه ای که شاه آنجاست قرمز بود، یعنی مهره ای از حریف روی صفحه است که شاه را کیش داده و بولین checkCalled که درابتدا فالس مقداردهی شده بود را ترو میکند
Void check4mate(Color color)	105	رنگ مات/پات کننده را ورودی می گیرد و برای تمام مهره های مات شونده تابع()possibleMoves را اجرا میکند، سپس با تابع ()pinخانه های قابل حرکت محدود میشوند(آچمز نبودن) شاه مات شونده جدا بررسی میشود زیرا دستور ()pin برای آن بی معنی است. نحوه ی بررسی به این صورت است که همانند قبل مختصات خانه ی شاه را بدست میآوریم، سپس ()kingActions را صدا میکنیم و

		خانه های غیر قابل دسترسی را با سبز و قرمز کردن زمینه غیرفعال میکنیم. حال hasKingMoves()را برای شاه مات شونده صدا میکنیم و زمینه ی خانه هایی که شاه میتواند برود به شرطی که خالی باشند یا پرباشند و مهره ی حریف در آن باشد را با خاکستری(بجای سبز) و خاکستری تیره(بجای قرمز) رنگ میزنیم. حالا مختصات شاه حریف را می یابیم و hasKingMoves() را با threatترو صدا میکنیم. و زمینه خانه هایی که شاه دیگر میتواند به آنها برود را نال میکنیم. حالا clearPossibleActions()را صدا میزنیم و زمینه خانه های سبز و قرمز (باقی مانده از صدا زدن خانه های دسترسی حریف) را نال کرده و زمینه های خاکستری را سبز و خاکستری تیره را قمز میکنیم. اینگونه حرکات ممکن شاه احتمالی را هم بدست اوردیم. برای هر مهره ی انتخاب شده پس از انجام اعمال گفته شده، زمینه ی کل مهره ها را بررسی میکنیم به طوری که اگر خانه ای بود که زمینه سبز یا قرمز (قابل دسترسی) داشت، بولین isMateرا فالس کند. با تمام شدن حلقه ی مهره ها، clearPossibleActions()را صدا میزنیم و زمینه ها را به حال اولیه برمیگردانیم در نهایت، درصورتیکه isMateهنوز ترو باشد، برنده را در پاپ آپ اعلام کرده و نیوگیم را صدا میزنیم
void coloredBGAction(int row, char col)	186	خانه ی کلیک شده را با مهره ای که در کلیک قبلی (که باعث رنگی شدن زمینه ی این خانه شده) پر میکنیم و مختصات کلیک قبلی را نال میکنیم. clearPossibleActions()را صدا میزنیم و سایر زمینه ها را به نال برمیگردانیم check4check()را صدا میزنیم.

		اگر checkCalledترو بود در پاپ آپ، شاهی که کیش شده را اعلام میکنیم. نوبت را عوض کرده و check4mate()را صدا میزنیم.
@Override **public void onClick(int row, char col)**	185	برای دسترسی به خانه صدا شده، هگزاگون سل targetCell را مساوی سل مختصات ورودی قرار میدهیم. اگر زمینه ی تارگت سل فیروزه ای بود یعنی باید سلکشن را ازآن برداریم که کافیست clearPossibleActions() صدا شود اگر سبز بود coloredBGAction() صدا میشود اگر قرمز بود ابتدا capture() سپس coloredBGAction() درغیر اینصورت زمینه ی آن بی رنگ(دیفالت) است. حالا اگر خود سل نال باشد، روی خانه ی خالی رندومی کلیک شده و زمینه زرد میشود اگر سل نال نباشد پس روی مهره ای کلیک شده و رنگ آن را در clickedColor تعریف میکنیم. اگر کلیکد کالر با ترن یکی بود یعنی نوبت مهره ی انتخاب شده رعایت شده و باید حرکاتی که میتواند انجام دهد را نمایش دهیم، اگر نبود در پاپ آپ اعلام میکنیم نوبت رعایت نشده اطلاعات مهره را در پراپرتی های nowBlue ذخیره میکنیم و در coloredBGAction() از آن استفاده میکنیم رنگ مهره ی حریف را مشخص کرده سپس تمام زمینه ها را خالی کرده و مهره ی انتخاب شده را فیروزه ای میکنیم و بقیه ماجرا را به possibleActions() و pin() پاس میدهیم.

		از حالات خارج میشویم و درصورتی که سربازی به انتها رسیده بود ()promote اش میکنیم چک برای کیش و برای مات/پات را صدا میزنیم و بالاخره ()onSave حرکت را در فایل بازی ذخیره میکند(اینطوری اگر قبل از پروموت کردن سرباز صفحه را ببندیم با باز کردن بازی به مشکلی بر نمیخوریم)
Color other(Color color)	195	رنگ حریف را پس میدهد
void clearPossibleActions()	272	اگر رنگ زمینه ای خاکستری بود ، سبز اگر خاکستری تیره بود، قرمز اگرنه زمینه را خالی میکنیم (تمایز برای شاه ایجاد شده)
void kingActions(Color enemy)	284	نوبت را موقتا به حریف میدهیم و چک2کالد را ترو میکنیم تا حالات پشتیبانی مهره ها از هم + کیش سرباز را در پاسیبل به حساب بیاورد و برای تمام مهره های حریف پاسیبل اکشنز را صدا میکنیم (شاهش هم جداگانه)
void capture(HexagonCell enemyCell)	309	*changesOnApp*.addRemovedPiece() صدا میشود. آن هم removedPiecesPanel.addPiece() را میخواند که یک آرایه1بعدی جدید به طول pieces.length+1 ایجاد میکند و با فور زدن از 0 تا طول-1 را با آرایه قبلی یکی میگذارد و استرینگ کالر ورودی گرفته شده را به انتها اضافه میکند.

void oneMove (int newRow, char newCol, Color enemy, boolean threat)	312	بررسی میکند که شاه 1 اگر شاه خودی است خانه هایی که از قبل توسط کینگ اکشنز رنگی نشدند و شاه اختیار حرکت به آن را دارد با خاکستری ها نشان دهد و 2 اگر شاه حریف است زمینه خانه هایی که به آن ها میتواند برود خالی شود
private void pin (int tryRow, char tryCol, String tryText, Color tryTestColor)	344	یک لیست ایجاد میکنیم و همه ی زمینه سبز ها را در آن میریزیم. سپس فور ایچ زمینه سبزی مهره ی ورودی داده شده را در آن قرار داده و چک کیش رو انجام میدیم و در صورتی که چک کالد فالس باشه یعنی مجازیم اونجا بریم و اونو به لیست دوم اضافه میکنیم. بیرون از فور میایم و لیست اول رو خالی میکنیم و همین کار رو برای زمینه قرمز ها تکرار میکنیم و توی لیست سوم میریزیمشون. آخرسر فور ایچ لیست دوم زمینه رو سبز و فور ایچ لیست سوم زمینه رو قرمز میکنیم
void possibleActions(int row, char col)	406	نوع مهره در سل.تکست کالر متمایز میشود، از این رو با ایف مهره ی انتخاب شده را با نوع مناسبش نیو میکنیم (مهره ی شاه محدودیت های مازاد دارد و آن را جدا از بقیه بررسی میکنیم که محدودیت ها را روی آن اعمال کنیم) در نهایت اگر مهره نال نبود، حرکات ولیدش را با piece.showValidMoves() صدا میزنیم
void onSave(File file)	589	پث فایلی که گرفتیمو ذخیره میکنیم تو متغیر ترنشو درمیاریم با B / W، دونه دونه همه ی خونه های جدولو اینطوری اضافه میکنیم به استرینگ مهره ها و یه اینت هم داریم تعداد اینارو حساب کنه "(" + col + row +","+text+"~"+textColor+ ")" با یه | جدا میکنیم بعد استرینگ حذفی ها رو اینطوری دونه دونه اضافه میکنیم(برای اینم تعداد شمار داریم): "(" +text+"~"+textColor+ ")"

		اینکه کیش شده یا نشده رو با C/D نشون میدیم حالا همه اینارو میچسبونیم به هم(اول تعداد مهره ها بعد # بعد هر استرینگ مهره بعد نوبت و کیش بعد | آخر هم تعداد و # و دونه دونه ی حذفیا) و میریزیم توی استرینگ اصلی و اون رو با فایلز.رایت استرینگ میریزیم توی اون پث فایل ورودی **همه ش توی ترای کچه و اگه مششکلی پیش اومد مثل IOExeption** بهمون میگه سیئ نشد
void onLoad(File file)		با فایلز.رید استرینگ میخونیم فایل رو، اول تعداد مهره ها رو نوشتیم پس تا # میره و اینتجر.پارس اینتش میکنه حالا یه فور اون تایی میزنه و هر سری استرینگی که قراره باهاش کار کنه رو میگیره سابستریگ اولین پرانتز باز تا بسته، بعد اون فرمی که توش ریختیم رو میخونه جدا میکنه با کر ات و اینتجر پارس اینت و استرینگ.ولیوآو هرچیز رو به همون فرمی که باید میخونه تبدیل میکنه و در نهایت قرار میدتشون با استفاده از تابع شما یعنی setCellProperties() اینکه گذشت کیش بودنو میخونه و آخر هم همینکارارو برای حذفیا میکنه
void onNewGame(File file)		همه چی رو دستی برمیگردونه به دیفالت نوبت رو سفید میکنه حذفیا رو خالی میکنه و کیش شدنو فالس میکنه

اصل این بود بقیه کلاسا نرمالن

یدونه ابسترکت پیس داریم که هرچی مشترک همه ی پیسا داشتن اعم از کانستراکتور و مختصات و رنگ و اینا توشه.

هر ساب کلسشم تنها فرقش اینه که ولید موو هاش همونطوریه که برای اون نوع مهره مجازه

Manager

اونا هم با مدل کردن دراومدن. و تقریبا همشون به قبل و بعد ستون f تقسیم شدن. مدل سازی اسبم هم خیلی خوب دراوردم. اسب رو لطفا نگاه کنید. راجع به بقیشون پیشنهادی ندارم

در نهایت یه نکته اضافی که واسه ی اون حالت پشتیبانی مهره ها از هم توی چک کردن کیش، میگم که توی ولیدموز اونجایی که لازم شد وایل رو قطع کنه چون به مهره ی خودی رسیده، در صورتی که چک2کالد ترو بود خود اون مهره هم پشتش رنگی شه، همین. برای سرباز هم ولید مووش اگه داره تهدید کیش رو نگاه میکنه کلا جلو رفتن رو بندازه دور فقط چپ و راستو رنگی کنه

امتیازی:

تغییر رنگ خانه ها

صدای حرکت مهره ها

یک فایل که ابسلوت پث برای یه آدیو از جنس wav هست رو توی متغیرمون میریزیم و بعد کلیپ اش رو میسازیم و .استارت میکنیم

گذاشتن ساعت برای حرکات

یک متود استاتیک (که همه جا بتونه صدا شه بدون نیاز به نیو کردن مای کلاک) به اسم استارت تایمر میسازیم و مقدار اولیه دقیقه و ثانیه ی سفید و سیاه رو بهش میدیم(یا دیفالت 0) بعد با اوورراید متد ران در کلاس تایمرتسک، حدی برای بازی تعریف میکنیم که در صورتی که به ده دقیقه رسید و بازی ادامه داشت بازی تمام شود و حریف ببرد. در غیر این صورت ثانیه ها را یکی یکی (با دیلی 1000 میلی ثانیه) زیاد کند و هروقت به 59 رسید، ثانیه صفر و به دقایق یکی اضافه شود. برای جدا کردن تایمر سفید و سیاه از پراپرتی ترن در کلاس اپلیکیشن استفاده کردم.

نمایش ساعت در بازی: با خواندن کلاس های گرافیکی نوشته شده، دو متد همانند ()setMessage ایجاد کردم و در اوورراید ()paintComponent در کلاس بوردپنل، آن هارا اضافه کردم و به رنگ بازیکن نمایش دادم

با تشکر از شما

سارا محمدی محمدی 402100504