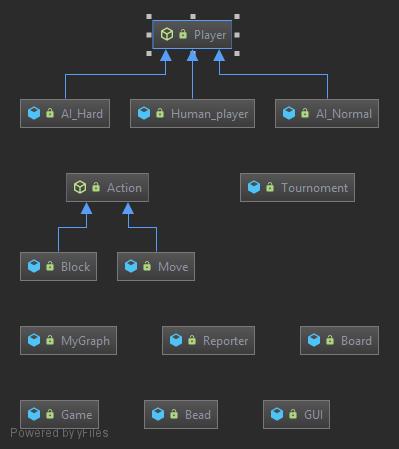
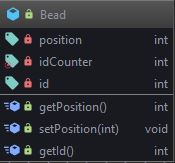
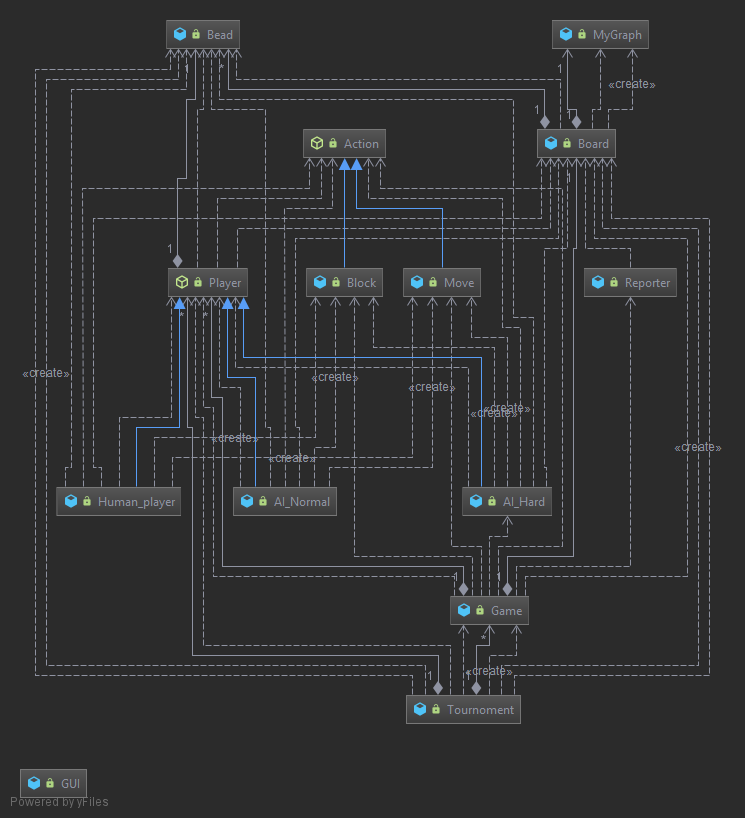
**JQuoridors**

در این پروژه کلاس های نوشته شده و آماده شده به زبان جاوا در راستای پیاده سازی انواع بازی کوریدور در انواع سیستم عامل ها بوده است ، برنامه نوشته شده در اصل به کلاس های پایه­یGame , move Bead , Player , Board , برای اجرای اولیه بازی و پیاده سازی قوانین اولیه بازی نیاز دارد با جلو تر رفتن برنامه و ایجاد مسائل جدید همچون دیوارگذاری و ساخت تورنومنت و هوش های مصنوعی متفاوت ( و در قسمت اختیاری گرافیکی ) نیاز به ساخت کلاس های دیگر همچون MyGraph , Block , AI\_hard , Reporter AI\_normal , Tournament , GUI پیدا کردیم که در خطوط بعد توضیحی مختصر از هرکدام داده خواهد شد.



**Bead**

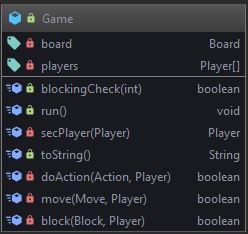
**در راه حل این مسئله به این نتیجه رسیدیم که بهتر است جدای از کلاس بازیکن و برد یک کلاس واسط وجود داشته باشد که نشان دهنده مختصات یک بازیکن است البته لازم به خاطرنشان است که هر مهره دارای یک شناسه خاص برای هر فرد است و به نوعی خاص به فرد است ، با توجه به نکات گفته شده قابل برداشت است که کلاس های بازیکن و بورد از مهره ها استفاده می کنند و قابلیت تغییر آن را دارند .**

**Move, Block**

**[ int pos, getPos (), setPos (),, int fpos , intspos , get&set spos&fpos]**

**این متد برای تغییر پوزیشن یک مهره و حرکت در برد است لازم به ذکر است که این کلاس از ابسترکت کلاس اکشن ارثبری می کند که فعلا اضافی بنظر می رسد اما در صورتی که بازی دارای چندین شیوه بازی باشد این متد باعث دسته بندی هرچه بهتر انواع حرکت خواهد شد به طور مثال با توجه به این که حرکت کردن و بلاک گذاری هردو زیرمجموعه "عمل" حساب می شوند هر دوباید از اکشن ارث بری کنند.**

**Game**



**در این کلاس با بهره گیری از کلاس های پلیر و بورد به انجام بازی می پردازیم یعنی در اصل حرکت های بازیکنان را مشخص می کنیم که از چه جنسی هستند و سپس با متد های بلاک و موو درست بودن این حرکت را ابتدا بررسی کرده و سپس درصورت درست بودن آن هارا انجام می دهد و درصورتی که نتواند ، اعلام می کند حرکت مورد نظر وجود ندارد. همچنین نوبت حرکت ها را نگاه میکند و بازیکن موردنظر را انتخاب می کند.**

**Reporter**

**کلاسی برای نمایش اجرای برنامه در ترمینال که برای کوتاه تر شدن گیم و فقط خروجی دادن استفاده شده است و در صورت اجرای گرافیکی استفاده خود را از دست میدهد.**

**Player**

**attributes:**

- String name // name of player

- int stick // number of stick for each player

-Bead bead // include position each player

- final int [] goalPlace // array [9] for goal place for each player

- int firstPlace // first place of each player

**Methods**

**در سازنده این کلاس مقادیر اسم و مهره و مکان اولیه پاس داده میشود . مقدار چوب ها برای هر بازیکن طبق قوانین بازی ۱۰ عدد در نظرگرفته شده است . و ارایه خانه های هدف طبق مکان اولیه هر بازیکن مقداردهی میشود و علت اینکار برای گسترش برنامه به بازی های ۴ نفره میباشد و گرنه میتوانستیم دو آرایه را به صورت مستقیم مقداردهی کنیم.**

LogicalComputing

**در این متد که یک متد ابسترکت است و پیاده سازی ندارد در حقیقت فقط در کلاس هوش مصنوعی استفاده و پیاده سازی میشود که کوتاه ترین راه را برای حرکت محاسبه میکند**

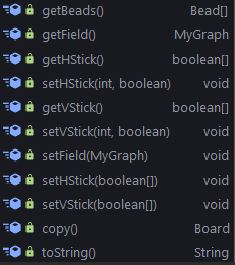
winCheck

**در این متد با روش گشتن در خونه های هدف هر بازیکن متوجه میشود که ایا بازیکنی به مقصد رسیده**

useStick

**در این متد و بقیه متد ها به نوعی ستر ها و گتر­ها هستند.**

**Board**



**این متد در اصل زمین بازی ما است و در آن چوب ها­ی عمودی و افقی و پوزیشن های زمین برای خود آرایه های مخصوص خود را دارند تا از آن ها بر ای پر بودن و تغییر جایگاه بتوانیم استفاده کنیم این کلاس از مهره ها برای پیدا کردن جای بازیکن ها و از بلاک ها برای پیدا کردن جیگاه چوب ها استفاده می کنیم.**

**AI`s**

**سناریو هوش مصنوعی سطح ۱(normal\_AI)**

**این هوش ابتدا پردازش هایی روی داده های موجود در صفحه بازی انجام میدهد و اطلاعات لازم برای انتخاب فعالیت بعدی برای انجام را داشته باشد. ابتدا با استفاده از الگوریتم BFS کوتاه ترین مسیر ممکن را برای برد پیدا میکند و خانه مجاور خود را در کوتاه ترین مسیر پیدا میکند.**

**سپس با احتمال ۶۵٪ به آن خانه حرکت میکند و در غیر این صورت در مجاورت بازیکن حریف (بالا پایین چپ راست) با احتمال های متفاوت بلاک میگذارد.**

**\*در صورتی که چوپ هوش مصنوعی تمام شده باشد احتمال حرکت کردن آن ۱۰۰٪ میشود.**

**احتمال جایی که بلاک میگذاریم به این صورت است که خانه ای که در جهت هدف های بازیکن است ۸۰٪ است و طرفین ۷.۵٪ و پشت ۵٪.**

**سناریو هوش مصنوعی سطح ۲(AI\_Hard)**

**این هوش در ابتدا پردازش های اولیه ای رو روی اطلاعات خام صفحه بازی انجام میده که نتایج آن خانه بعدی برای حرکت در کوتاه ترین مسیر - مقدار کوتاه ترین مسافت خود هوش و بازیکن حریف در مرحله بعدی پردازش به دنبال بهترین جای ممکن برای قرار دادن بلاک هستیم که این هوش برای انتخاب این بلاک تمام بلاک های ممکن را برسی میکنه و بلاکی با گذاشتن آن کمترین مسافت ممکن برای رسیدن حریف به مقصد را بیشتر از همه افزایش میدهد را انتخاب میکند. در مرحله بعد اگر کوتاه ترین مسافت ممکن برای برای خود هوش از بازیکن حریف کمتر باشد حرکت میکند و در غیر این صورت بلاک میگذارد.**

**\*اگر بلاک هایش تمام شده باشد حرکت میکند**

**\*اگر به هر دلیل نتواند بلاک را قرار دهد حرکت میکند (جای بهینه ای برای گذاشتن بلاک وجود نداشته باشد).**

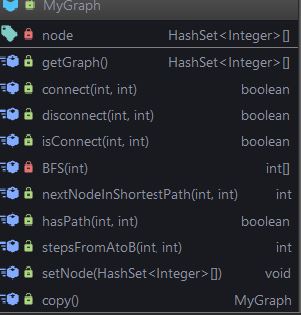
**سناریو هوش مصنوعی سطح ۳ (AI\_Insane)**

**این هوش در ابتدا پردازش های اولیه ای رو روی اطلاعات خام صفحه بازی انجام میده که نتایج آن خانه بعدی برای حرکت در کوتاه ترین مسیر - مقدار کوتاه ترین مسافت خود هوش و بازیکن حریف در مرحله بعدی پردازش به دنبال بهترین جای ممکن برای قرار دادن بلاک هستیم که این هوش برای انتخاب این بلاک تمام بلاک های ممکن را برسی میکنه و در هر مرحله مقدار مسافتی که به خودش و حریف اضافه میکند را با هم مقایسه میکند و اختلاف آنها را به دست می آورد و بلاکی را انتخاب می کند که بیشترین اختلاف را داشته باشد و مقدار مسافتی که به هوش اضافه شده کمتر از مسافتی که به حریف اضافه شده باشد.**

**در مرحله بعد اگر کوتاه ترین مسافت ممکن برای برای خود هوش از بازیکن حریف کمتر باشد حرکت میکند و در غیر این صورت بلاک میگذارد.**

**\*اگر بلاک هایش تمام شده باشد حرکت میکند**

**\*اگر به هر دلیل نتواند بلاک را قرار دهد حرکت میکند (جای بهینه ای برای گذاشتن بلاک وجود نداشته باشد).**



**MyGraph**

**این کلاس یک مدل سازی از گراف با یکسری از امکانات محدود گراف که از آنها در پروژه استفاده کردیم است.**

**که شامل یک آرایه از مجموعه است که هر ایندکس از آرایه یک راس گراف میباشد که شامل یک مجوعه از ایندکس راس هایی که آن متصل است میباشد.**

**فعالیت های این کلاس شامل موارد زیر میباشد:**

**-قطع و وصل کردن راس های گراف و چک کردن وصل بودن دو راس**

**-پیدا کردن راس بعدی برای پمایش در کوتاه ترین مسیر از راس A به راس B**

**-چک کردن وجود مسیر بین دو راس**

**-دریاف مقدار کوتاه ترین مسیر از راس A به راس B**

**Tournament**

**با توجه به این که ممکن است چندین بازی همزمان از ما خواسته شود و یا نیاز داشته باشیم یک جام از بازی ها درست بکنیم نیاز به کلاس تورنومنت خواهیم داشت این کلاس آرایه ای از گیم درون خودش نگاه می­دارد و با کمک دو متد به صورت رندوم و یا ترتیبی بازی ها را دریافت کرده و اجرا می­کند این متد از نظر اجرا بسیار شبیه به خود متد گیم است با این تفاوت که مجموعه ای از گیم ها اجرا می­شود.**

**Developers**



**امیرعلی خانه عنقا : برنامه نویسی تورنومنت ها ، اجرای ساختار dfs و یکی کردن تکه های پروژه**

**ایمان سلطانی : برنامه نویسی هوش های مصنوعی و ساختار bfsو کلاس بندی اولیه پروژه**

**مهدیه نعیمی :فرمول بندی بورد و نحوه استقرار مهره ها در زمین ،ساختار بندی اولیه پروژه و بلاک­ها**

