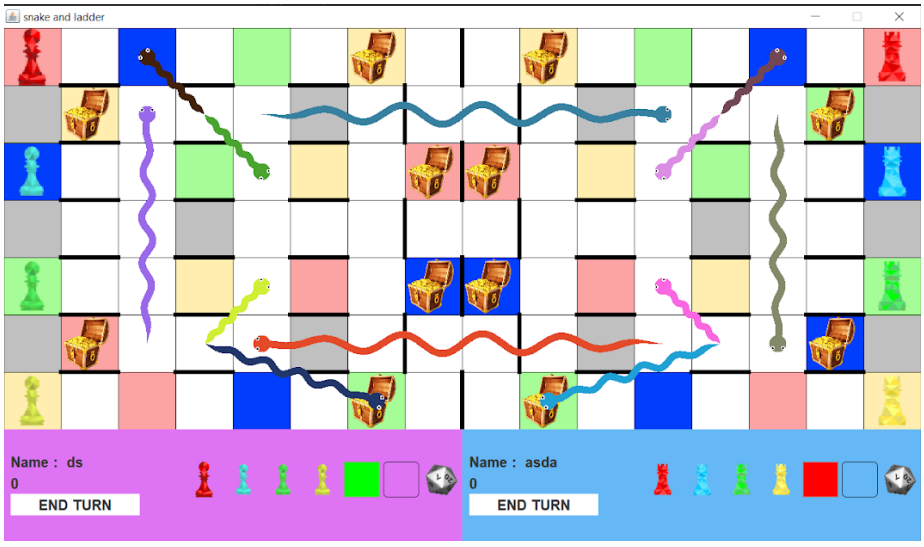
به نام خدا

پروژه درس مبانی برنامه نویسی

بخش اول



پاییز 1399– دانشکده علوم ریاضی

دانشگاه صنعتی شریف

تیم طراحی :

سید امیر محمد سادات شکوهی

سید محمد ترابی

سید عرفان موسویان

محمد مهدی زارع

هلیا نیاپرست

کسری خوشجو

محمدعلی علما

معرفی بازی :

تیم طراحی تمرین و پروژه درس مبانی برنامه نویسی، بعد از خستگی نیم ترم اول ، دیگر رمقی برای طراحی هزاران تمرین و تست کیس برای دانشجویان درس ندارند. بنابراین برای آنکه دانشجویان درس در ادامه ترم از وجود تمرین بی‌بهره نمانند و بتوانند مطالب درسی را به خوبی بیاموزند، به ناچار تصمیم گرفت که درخواست یک پروژه را به شرکت بین المللی "کدج" بدهد.

شرکت "کدج" که پیش زمینه خوبی در طراحی تمرین و پروژه نداشت، شرایط بسیار سختی را در پیش گرفت و پروژه بسیار سنگین و سختی را طراحی کرد و در اختیار تیم درس گذاشت. حتی به دلیل کمبود وقت، فقط قسمتی از پروژه را به تیم تحویل داد و قرار شد به مرور زمان بقیه پروژه را نیز در اختیار آنها قرار دهد.

تیم درس پس از بررسی و برگزاری جلسات متعدد، به این نتیجه رسیدند که بازی طراحی شده توسط شرکت "کدج" برای دانشجویان بسیار سنگین خواهد بود، فلذا تصمیم گرفتند که بخشی از پروژه را تکمیل کنند و مابقی پروژه در اختیار دانشجویان قرار بگیرد تا آن‌ها آن را کامل کنند.

بازی طراحی شده توسط شرکت " کدج " ، بازی BP Wars : Snake & ladder بود که از جهت‌های بسیاری شبیه به بازی خاطره انگیز مارو‌پله است. اما اعضا خبیث شرکت "کدج" بازی مذکور را تغییر داده‌اند و آن را شبیه به بازی‌های استراتژیک کرده اند.

در قسمتی از توضیحاتی که شرکت "کدج" برای اعضا تیم درس فرستادند آمده است که : " ... در این بازی دونفره، هر بازیکن چهار مهره خواهد داشت. هر بازیکن ابتدا باید تاس بیاندازد و سپس به اندازه عدد روی تاس، یکی از مهره های خود را حرکت دهد. در صفحه بازی، هیولاهای خطرناکی وجود دارند که اگر آنها را بیدار کنید، به ضرر خودتان(و چه بسا سایر بازیکنان) خواهد بود، همچنین برای مقابله با این هیولاها ، چندین گنج در بازی وجود دارد که با بدست آوردن آن‌ها، شاید بتوانید بهتر با هیولاها یا حریفان خود مقابله کنید. لازم به ذکر است ما به هیچ عنوان اعتقادی به پله نداریم ... "

معرفی فاز اول

هدف این بخش پیاده سازی دیتا مدل‌های بازی و بخشی از منطق بازی است. بخش اول پروژه‌ی شما پایه و اساس بخش‌های دیگر پروژه است.

اهداف بخش اول

- طراحی و نگهداری دیتا مدل‌های یک برنامه به صورت شی‌گرا
- ذخیره‌ی داده و کار با فایل

ارزیابی

- در زمان تحویل حضوری، کدهای آپلود شده در کوئرا، مورد بررسی قرار می‌گیرند.
- این بخش به همراه دو بخش بعدی پروژه (بخش ۲ و ۳) تحویل داده می شود.

روش‌تحویل

با اینکه این بخش به همراه دو بخش بعدی آن تحویل گرفته می‌شود اما لطفا به موارد زیر توجه کنید:

- هر بخش پروژه به طور جدا و به صورت حضوری ارزیابی می شود.
- در ارزیابی یک بخش، کدها بخش های قبلی در نظر گرفته نمی شوند(مثلا اگر بخش یک اشکال داشته باشد و نمره کامل دریافت نکند، این ربطی به بخش دو ندارد. البته اگر مشکل بخش یک روی کارکرد بخش دو اثر بگذارد در ارزیابی تاثیر می گذارد. در واقع هر بخش به طور کاملا جدا بررسی می شود و فقط کارکرد آن مدنظر است. اما اگر این کارکرد بخاطر بخش های قبلی، مناسب نباشد نمره بخش جدید هم، طبیعتا کم می شود).

موارد غیر مجاز

- عدم تسلط کافی بر کد پروژه
- شباهت بیش از حد دو یا چند پروژه
- واگذاری کامل یا بخشی از پروژه به شخصی دیگر
- کپی کردن کل یا بخشی بزرگی از کد(سورس کد) از منابع و پروژه های موجود در اینترنت

توجه کنید که مشورت و پرسش و پاسخ از دستیاران آموزشی یا دوستان مصداق تقلب نیست اما در اختیار گذاشتن بخش یا کل کد شما چه به صورت Pseudo Code یا Exact Code Implementation غیر مجاز است.

همچنین استفاده از اینترنت و کدهای موجود در آن مشکلی ندارد (البته باید در بخش توضیحات پروژه آن را بنویسید) اما نباید حجم زیادی از پروژه شما از منابع دیگر کپی شده باشد. سعی کنید حد تعادل را رعایت کنید.

رخ دادن این اتفاق‌ها برای هیچ فردی قابل پذیرش نیست و در صورت بروز هر کدام از این اتفاق‌ها ممکن است هر تصمیمی در رابطه با ارزیابی فرد گرفته شود.

در صورتی که یکی از اتفاقات بالا رخ دهد، اگر دانشجویان مسئول قبل از تشخیص تیم ارائه درس، مسئله را مطرح کنند فقط نمره مربوط به بخش اعلام شده تحت تاثیر قرار خواهد گرفت.

در صورتی که یکی از موارد بالا توسط تیم درس، قبل از اعلام شخص، مشاهده شود؛ تیم از فرد (افراد) درخواست خواهد کرد که در این رابطه توضیح دهند و در صورت قابل قبول نبودن توضیح، فرد موفق به گذراندن درس نخواهد شد.

تمامی قسمت های مربوط به گرافیک بازی از قبل آماده شده است و نیازی نیست که شما کد مربوط به گرافیک را بلد باشید. کد های مربوط به بخش گرافیک در پکیج گرافیک قرار دارد و نباید هیچ تغییری از سمت شما اعمال شود.

مجددا توجه کنید که به کد های گرافیک نباید دست بزنید و هیچ نیازی به دانش گرافیک ندارید.

بخش اول - Game Models & Basic Game Logic

بخش اول پروژه شما به بخش‌های زیر تقسیم می‌شود:

1. **مدل‌های بازی**
2. **منطق بازی**

دیتا مدل‌ها، مدل‌های انتزاعی هستند که داده‌های دنیا واقعی را نمایش می‌دهند و رابطه بین آن‌ها را تعریف می‌کنند. به عبارت دیگر، دیتا مدل‌ها، اشیا بازی شما هستند که توسط قسمت‌های مختلف برنامه استفاده می‌شوند. وظیفه اصلی شما در این بخش طراحی و مدیریت این مدل‌ها است. در این بخش شما باید سیستم شی‌گرا از داده‌های بازی را کامل کنید.

لطفاً دقت کنید که در **این فاز**، عملکرد کامل بازی مطرح نیست ولی کارکرد درست و کامل بخش‌هایی که جلوتر تعریف شده‌اند اهمیت دارد و مورد ارزیابی قرار می‌گیرد.

کد اولیه پروژه را از این لینک دریافت کنید و آن را باز کنید .

https://github.com/samssh/snake_and_ladder

- **توجه داشته باشید که در فاز های بعدی، تغییراتی در بازی اعمال خواهد شد. بنابراین سعی کنید با ساده سازی قسمت های مختلف و تکه تکه کردن انجام هر کاری(استفاده از توابع) ، بازی خود را طوری طراحی کنید که در صورت تغییر در قسمت های بازی، نیازی به تغییرات بسیار گسترده در کد نداشته باشید.**

- **همچنین توجه کنید که میتوانید از کلاس ها و یا توابع آماده خود جاوا نیز، در صورت نیاز استفاده کنید.**

1. منطق بازی

- در ابتدای بازی، اسم دو بازیکن از کاربر گرفته میشود و این اسامی به ترتیب به منطق بازی منتقل میشود (این کار به صورت خودکار در گرافیک صورت می گیرد)
- هر بازیکنی، باید یک فایل داشته باشد. با گرفتن اسم بازیکن، اگر فایل بازیکن وجود دارد، بازیکن را از روی بازیکن لود کنید. اگر وجود ندارد برای او یک فایل نیز بسازید.

- هر بازیکن چهار مهره با رنگ های آبی، قرمز، زرد، سبز دارد.
- زمین بازی یک مستطیل با 7 سطر و 16 ستون است.
- توجه داشته باشید مختصات بازی اندکی متفاوت است: یعنی بردار x در راستای عمودی و بردار y در راستای افقی است .
- هر خانه دارای رنگی خاصی است و ممکن است در بعضی از خانه‌ها جایزه یا مار باشد.
- در ابتدای بازی هر دو بازیکن باید اعلام آمادگی کنند تا بازی شروع شود.
- در هر نوبت یک بازیکن حق دارد یک بار تاس بریزد. و به مقدار عددی که آورده است، یکی از مهره هایش را در یک جهت دلخواه در صورت معتبر بودن حرکت انتقال دهد. (حرکت صرفا به صورت عمودی و یا افقی است)
- هیچ مهره‌ای نمی‌تواند در خانه های سیاه توقف کند.
- همه مهره‌ها می توانند به خانه های سفید بروند.
- هر مهره می‌تواند به خانه های همرنگ برود. ولی نمی‌تواند در خانه های ناهمرنگ به جز سفید توقف کند.(مثلا مقصد مهره زرد، نمی تواند خانه آبی باشد.)
- مهره در طول مسیری که طی می‌کند نباید از دیوار ها رد شود.
- هیچ مهره‌ای نمی تواند وارد خانه‌ای شود که مهره دیگری در آن است.
- اگر هیچ مهره‌ای نتواند با اندازه مقداری که تاس آماده حرکت کند از آن بازیکن 3 امتیاز کسر می‌شود و نوبت بازیکن بعدی می‌شود.
- اگر مهره‌ای وارد خانه‌ای شود که سر یک مار در آن باشد این مهره به خانه‌ی انتهای مار می‌رود و 3 امتیاز کسر می‌شود. اگر خانه‌ی انتهای مار پر بود مهره انتقال نمی‌یابد.(اما امتیاز او کسر می‌شود)
- اگر خانه‌ای حاوی جایزه بود، مهره وارد شده از آن جایزه استفاده می‌کند و جایزه از بین نمی‌رود و میتوان از آن مجددا استفاده کرد.
- اگر بازیکنی با تاس خود 6 بیاورد، 4 امتیاز کسب می کند. (تاس 6، جایزه ریختن دوباره تاس **ندارد**)
- اگر مهره ای وارد خانه همرنگ خودش شود، 4 امتیاز کسب می‌کند.
- بازی بعد از 30 بار تاس انداختن تمام می‌شود.
- بعد از اتمام بازی، باید فایل هر بازیکن را آپدیت کنید.(نحوه پیاده سازی به عهده خود شماست)
- همچنین، باید **یک** فایل داشته باشید که اطلاعات تمام بازی های انجام شده را در آن ذخیره کنید . این اطلاعات شامل اسم هر دو بازیکن، امتیاز هر کدام و برنده بازی است . نحوه پیاده سازی این کار ، به عهده خود شماست.

2.مدل‌های بازی

در ادامه ساختار مدل های بازی را توضیح می دهیم. هر تابعی که نیاز به توضیحات تکمیلی داشته باشد را اینجا توضیح می‌دهیم . اگر تعریف تابعی واضح باشد یا توضیحات آن در فایل همان کلاس کافی باشد، توضیحی نداده‌ایم.

- **گیم استیت:**
  - فیلدهای اصلی:
    - برد
    - بازیکن ها
    - شماره دور (چه تعداد از دورهای بازی گذشته است)
  - توابع:
    - گرفتن بازیکن فعلی
      - بازیکنی که در حال حاضر نوبت او است را برمی‌گرداند.
    - دور بعد
      - از این تابع برای خاتمه دادن به نوبت یک بازیکن و ادامه یافتن بازی توسط بازیکن دیگر استفاده می‌شود .
    - گرفتن بازیکن بر حسب شماره آن‌ها

- **برد:**
  - فیلد ها:
    - لیست خانه‌ها
    - لیست انتقال دهنده‌ها
    - لیست خانه‌هایی که مهره‌ها در اول بازی آنجا هستند.
    - لیست دیوارها
  - توابع مهم:
    - گرفتن یک خانه بر حسب مختصات آن

- **بازیکن:**
  - فیلد ها:

- اسم
- امتیاز
- لیست مهره ها
- تاس
- بازیکن حریف
- آیدی
- شماره بازیکن
- اعلام آمادگی کرده است یا خیر
- در این نوبت تاس ریخته است یا خیر
- مقداری که تاس آورده است (در واقع تعداد حرکات باقی مانده توسط مهره moveLeft است)
- مهره ای که برای حرکت دادن انتخاب کرده است

- توابع:
  - استفاده کردن از جایزه
  - آیا توانایی حرکت با مهره های خود را دارد؟
    - از این تابع زمانی استفاده می‌شود که میخواهیم چک کنیم آیا بازیکن با تاسی که ریخته است، توانایی حرکت حداقل یکی از مهره های خود را دارد یا نه.

  - ریست کردن فیلدهای مربوط به یک دور
    - زمانی که نوبت بازیکن تمام می‌شود، باید تعدادی از فیلد های او بازنشانی شوند.

- **مهره:**
  - فیلد ها:
    - خانه فعلی آن مهره
    - رنگ آن مهره
    - بازیکن صاحب آن مهره
    - آیا این مهره انتخاب شده است یا خیر
  - توابع:
    - آیا توانایی رفتن به یک خانه‌ی خاص را دارد؟
      - با این تابع می‌توانید چک کنید که آیا مهره با توجه به مکان فعلی‌اش، توانایی رفتن به یک خانه دیگر (که به عنوان ورودی تابع داده می‌شود) را دارد یا نه.
    - جابه‌جا کردن آن مهره
      - مکان فعلی مهره را تغییر دهید، و مکان قبلی مهره را نیز در صورت نیاز پاکسازی کنید.

- **خانه:**

- فیلدها:
  - رنگ
  - مختصات
  - انتقال دهنده
  - خانه های مجاور
  - جایزه
  - مهره‌ی داخل خانه
  - خانه‌های همسایه که باز هستند.
  - خانه‌های همسایه
- توابع:
  - آیا یک مهره‌ی خاص می تواند وارد این خانه شود؟
    - باید چک کنید که مهره‌ای که به عنوان ورودی تابع داده می‌شود، طبق قوانین بازی می‌تواند وارد این خانه شود یا نه.

- **دیوار:**
  - فیلدها:
    - خانه‌ی اول و دوم

- **انتقال دهنده:**
  - فیلد ها:
    - خانه مقصد و مبدا
  - توابع:
    - انتقال مهره

- **جایزه:**
  - فیلد:
    - خانه‌ای که جایزه در آن قرار دارد
    - مقدار امتیازی که به بازیکن اضافه می‌کند
    - مقدار افزایش شانس x آمدن تاس
  - توابع:
    - استفاده توسط یک مهره

- **تاس:**
  - فیلد:
    - فیلدهای مربوط به تاس را خودتان برحسب نیاز باید بنویسید و مقداردهی کنید.
  - توابع:
    - انداختن تاس
    - شانس آمدن یک عدد عوض شود.

- مشخصات تاس (اینکه هر شماره چه احتمالی دارد)

- **رنگ**:
  - توجه داشته باشید که شما نیازی به تغییر این مدل **<u>ندارید</u>** و صرفا باید از آن استفاده کنید.
  - برای استفاده از این مدل به دو روش می توانید عمل کنید :
    - استفاده از تابع valueOF : یک رشته به عنوان ورودی می‌گیرد و شیء ای با این اسم در داخل این کلاس را برمی‌گرداند .توجه داشته باشید که به بزرگی و کوچکی حروف حساس است.(نیازی به دانستن نحوه پیاده سازی این کلاس نیست)
      - مثال :

```
Color c = Color.valueOF("BLUE");
```

    - استفاده به صورت مستقیم :
      - مثال :

```
Color c = Color.BLUE;
```

- **توجه داشته باشید که ممکن است کلاس های دیگری نیز به اسم Color در جاوا وجود داشته باشد. شما باید از کلاس Color موجود در پکیج model استفاده کنید، در غیر این صورت بازی شما با مشکل مواجه خواهد شد.**

پکیج بندی و کلاس‌های بازی

به هیچ عنوان در کلاس‌ها، فایل‌ها و پکیج‌هایی که در ادامه گفته خواهد شد، تغییراتی ایجاد نکنید.

1. پوشه resources :

در این پوشه، فایل‌های کانفیگ برنامه، عکس ها و بورد بازی قرار دارد . بازی شما برای اجرا شدن (چه از نظر منطق و چه از نظر گرافیک) نیاز به این فایل ها خواهد داشت.

نیازی به ایجاد تغییر در فایل های **از پیش طراحی شده** این پوشه **نیست**. اما شما باید توانایی ساخت، ذخیره و لود کردن برخی فایل ها را از این پوشه را داشته باشید (توضیحات در قسمت مربوطه داده خواهد شد.)

2. پکیج model :

در این پکیج کلاس‌های مربوط به مدل‌های بازی قرار گرفته است. در صفحات قبلی آنها را توضیح دادیم .

3. پکیج logic :

در این پکیج کلاس‌های مربوط به منطق بازی قرار گرفته است.

برای ارتباط دو بخش منطق و گرافیک بازی ، دو کلاس LogicalAgent (موجود در پکیج Logic) و کلاس GraphicalAgent(موجود در پکیج Graphic) وجود دارند. این دو کلاس واسطی بین منطق و گرافیک بازی خواهند بود . اگر کاربر در محیط گرافیکی برنامه عملی را انجام دهد، این عمل از طریق کلاس GraphicalAgent به توابع کلاس LogicalAgent منتقل میشود . قسمتی از کلاس LogicalAgent نیز طراحی شده است، وظیفه شما کامل کردن توابع این کلاس است . مقادیر خروجی از این توابع ، به صورت خودکار به گرافیک بازی منتقل و سپس نمایش داده خواهد شد(به شرطی که که شما منطق بازی را به درستی طراحی کرده باشید)

- **LogicalAgent:**

- فیلد:
  - ModelLoader
  - GraphicalAgent
  - GameState
- توابع:
  - آماده شدن بازیکنان
  - انتخاب خانه
    - هر زمان که کاربر در گرافیک بازی روی خانه‌ای با مختصات x,y کلیک کند این تابع به صورت اتوماتیک صدا زده می‌شود. شما باید :
      - اگر مهره ای از قبل انتخاب شده باشد، چک کنید که می‌توانید آن را به خانه‌ی جدید (با مختصات داده شده) منتقل کنید یا نه و در صورت امکان آن را منتقل کنید.
      - اگر مهره‌ای از قبل انتخاب نشده باشد، اگر در آن خانه مهره ای وجود دارد، آن را انتخاب کنید.
  - چک کردن پایان بازی
    - در این تابع باید شرایطی را چک کنید که آیا بازی تمام می‌شود یا نه. قسمتی از این تابع نوشته شده است. شما آن را تکمیل کنید (شروط لازم برای پایان بازی را اصلاح کنید. شرایط در صفحات قبلی توضیح داده شده است.) همچنین شما باید برنده را مشخص کنید .
  - انداختن تاس:
    - زمانی که کاربر روی دکمه تاس "کلیک چپ" می‌کند، به طور اتوماتیک این تابع صدا زده می‌شود . شما باید با متصل کردن این تابع به لاجیک بازی، برای بازیکن مورد نظر تاس بیندازید و برخی مدل های بازی را آپدیت کنید.
  - گرفتن مشخصات تاس
    - زمانی که کاربر روی دکمه تاس "کلیک راست" میکند، به طور اتوماتیک این تابع صدا زده می‌شود . شما باید با متصل کردن این تابع به لاجیک بازی خود، یک رشته را به این تابع بدهید و آن رشته را برگردانید(نیازی به چک کردن مراحل بعدی مرتبط با گرافیک نیست)

- **BoardBuilder:**
  - کانستراکتور:
    - به عنوان ورودی کانستراکتور، یک رشته به شما داده می‌شود. شما باید بتوانید این رشته را بخوانید و آن را به برد بازی تبدیل کنید.
  - فیلد:
    - فیلد های مورد نیاز را خودتان اضافه کنید.
  - توابع:
    - ساختن

- با استفاده از این تابع می‌توانید برد بازی خود را بسازید و از آن استفاده کنید .
- میتوانید توابع دیگری را نیز برای خوانایی کد خود به این کلاس اضافه کنید.

- **ModelLoader :**
-
  - فیلد:
    - سه فایل boardFile , playersDirectory , archiveFile
      - این سه فیلد در داخل کانستراکتور مقداردهی شده است و نیازی به انجام تغییرات توسط شما نیست.
    - در صورت نیاز میتوانید فیلد های دیگری نیز اضافه کنید.

  - توابع:

    - لود کردن برد بازی
      - با استفاده از فایل boardFile باید بتوانید یک برد بسازید. (میتوانید از کلاس BoardBuilder برای این منظور استفاده کنید.) قسمتی از این تابع نوشته شده است ، شما باید تغییرات لازم را در قسمت "try" این تابع بنویسید .
    - لود کردن پلیر
      - باید برای هر کدام از بازیکن ها، یک فایل داخل پوشه ی playersDirectory وجود داشته داشته باشد که این فایل شامل اسم بازیکن و امتیاز او(مجموع امتیاز تمام بازی هایی که تاکنون انجام داده) خواهد بود . قسمتی از کد این تابع نوشته شده است. شما این تابع را تکمیل تر کنید . (کد های خود را در ساختار "try" بنویسید)
        - شما مجازید هرگونه که میتوانید این کار را انجام دهید. نحوه پیاده سازی این کار را با سلیقه خود انجام دهید. این کار را به گونه ای انجام دهید که توانایی ذخیره و سپس لود کردن این فایل ها را داشته باشید.
    - گرفتن فایل بازیکن
      - در ورودی به شما اسم بازیکن داده شده است. شما باید فایل مربوط به این بازیکن را بخوانید و آن را برگردانید.
    - سیو کردن بازیکن

- برای ذخیره یا آپدیت فایل بازیکنان میتوانید از این تابع استفاده کنید. در ورودی به شما یک بازیکن داده میشود . یا فایل مربوط به او را آپدیت کنید یا برای او یک فایل بسازید(در صورتی که فایل نداشته باشد)
  - آرشیو
- با استفاده از این تابع ، یک فایل شامل اطلاعات بازی انجام شده توسط این دو بازیکن (شامل اسم، امتیاز هر کدام و برنده بازی) بسازید و آن را ذخیره کنید. نحوه پیاده سازی این عملیات به عهده خود شماست.

4. پکیج گرافیک :

در این پکیج و کلاس های داخل آن، هر آن چیزی که برای نمایش بازی به صورت گرافیکی نیاز است، طراحی شده است. کارکرد صحیح این بخش، منوط به کارکرد صحیح منطق بازی است که توسط شما طراحی خواهد شد.

نیازی به ایجاد تغییر در هیچ یک از کلاس های این پکیج <u>نیست</u>.

مهمترین کلاس این پکیج GraphicAgent است که وظیفه آن (در کنار LogicalAgent) ایجاد واسطه ای بین منطق و گرافیک بازی است.

5. پکیج util :

سایر کلاس های برنامه که برای اجرای صحیح بازی بسیار مورد نیاز هستند، در این پکیج قرار دارند .

نیازی به ایجاد تغییر در هیچ یک از کلاس های این پکیج <u>**نیست**</u>.

توضیحات فایل board

- **توجه داشته باشید که این برد یک برد نمونه است و تمام محتویات آن (شامل رشته ها و اعداد) نمونه هستند و ممکن است در هنگام تحویل پروژه،**

از برد دیگری(با همین فرمت) استفاده شده باشد. بنابراین نحوه خواندن و نوشتن در فایل را به صورت منعطف تری طراحی کنید.

در این فاز، فایل برد شامل 5 قسمت است :

1. اطلاعات مربوط به رنگ هرکدام از خانه های بازی نوشته شده است .(راهنمایی : می توانید با استفاده از اسکنر اطلاعات را بخوانید.) در هر خط، تعدادی رشته که معادل رنگ هر خانه است، آماده است. شما باید این رشته ها را به ترتیب بخوانید و از روی این رنگ ها یک خانه های بازی را بسازید (از کلاس Color و توابع آن که چند صفحه قبل توضیح داده شد، برای مقداردهی رنگ هر خانه استفاده کنید)

2. اطلاعات مربوط به خانه های شروع بازی داده شده است . در هر خط سه عدد x , y , i به ترتیب آماده است که به ترتیب (از راست به چپ) بیانگر بازیکن i و مختصات x , y مربوط به آن خانه است.
3. اطلاعات مربوط به دیوار ها آمده است. در هر خط اطلاعات مربوط به یک دیوار آماده است که اعداد داده شده به ترتیب مربوط به مختصات خانه اول و دوم است که دیوار بین آنها قرار گرفته. این اعداد به ترتیب x1,y1,x2,y2 هستند.
4. انتقال دهنده ها : اطلاعات مربوط به انتقال ها آمده است. در هر خط اطلاعات مربوط به یک دیوار آماده است که اعداد داده شده به ترتیب مربوط به مختصات خانه اول و دوم است که انتقال دهنده بین آنها قرار گرفته. این اعداد به ترتیب x1,y1,x2,y2 هستند.
   a. **توجه داشته باشید که ترتیب این دو خانه مهم است و انتقال دهنده از خانه اول شروع میشود و بازیکن را به خانه دوم منتقل میکند.**
5. جوایز : اطلاعات مربوط به جوایز آمده است. در هر خط اطلاعات مربوط به یک جایزه آماده است که اعداد داده شده به ترتیب

x , y , s , c , n

هستند . این بدین معناست که این جایزه، در خانه ی x,y قرار گرفته ، به میزان s امتیاز بازیکن را تغییر میدهد و همچنین شانس آمدن عدد n در تاس را به میزان c تغییر میدهد.

- **توجه داشته باشید که تضمین میشود تمام مختصات داده شده معتبر هستند.**