|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| **UNO Guide**  ***"Expect the best, plan for the worst, and prepare to be surprised." - Denis Waitley*** |
| |  |  |  | | --- | --- | --- | | Keivan Ipchi Hagh | 4/11/20 | AP Project – Part 3 | |

Table of Contents

[مشخصات ابتدایی پروژه 2](#_Toc37844138)

[دستور العمل (نحوه تعامل با بازی) 2](#_Toc37844139)

[نوع و کاربرد کلاس های استفاده شده 3](#_Toc37844140)

[الگوریتم موتور بازی 3](#_Toc37844141)

[الگوریتم هوش مصنوعی بازی 3](#_Toc37844142)

[شیء گرا بودن و استفاده از مفهوم Encapsulation, Encapsulation, Inheritance 3](#_Toc37844143)

[نمایش 4 نفره vs 7 نفره 4](#_Toc37844144)

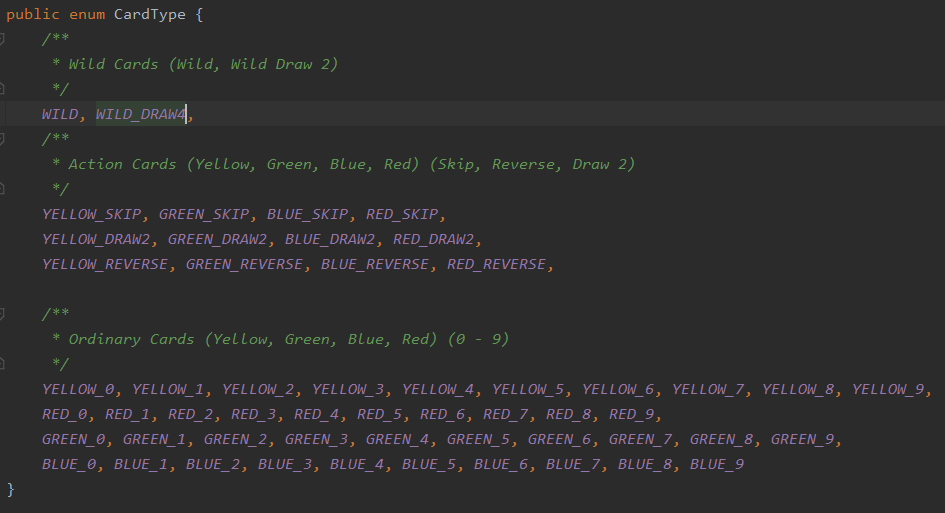
[چند نکته مهم! 4](#_Toc37844145)

# مشخصات ابتدایی پروژه

|  |  |
| --- | --- |
| عنوان پروژه | بازی هفت خبیث |
| استاد درس | حسین زینلی |
| تاریخ تحویل | بیست و هفت فروردین 1399 |
| نویسنده | کیوان ایچپی حق – 9831073 |

# دستور العمل (نحوه تعامل با بازی)

* ابتدا مشخص میکنید بازی چند بازیکن خواهد داشت. توجه کنید اگه 4 بازیکن انتخاب کرده باشید صفحه گرافیکی نسبتا خفن تری خواهید داشت 😉
* سپس مشخص میکنید تعداد بازیکن های AI چند تاست؛ تمام بازیکن ها میتوانند AI باشند و یا هیچ بازیکنی AI نباشد.
* اسم بازیکن های خود را داده و اسم بازیکن های AI اتوماتیک تولید میشوند.
* انتخاب های کاربر از بین کارت ها با syntax زیر است:



# نوع و کاربرد کلاس های استفاده شده

پروژه از تعدادی *class* تشکیل شده که هرکدام مسئول انجام بخشی از عملیات مربوط به بازی است. بازی از یک بورد تشکیل شده که در *Board* به صورت ماتریس ذخیره شده و تعدادی تابع مسئول پرینت کردن این ماتریس به صورت گرافیک عادی یا گرافیک یکم باحال تر (بنابر خواسته کاربر) است. هر عضو این لیست شیئی از نوع *“Card”* است. هر یک از این عضوها، دارای مقدار، رنگ، نوع است. مقدار یک *ASCCI CODE* برای نشان توع کارت، نوع در اصل *Enum بوده که نوع و رنگ کارت را مشخص میکند. Player ها که هریک، اطلاعات هر بازیکن نظیر اسم و امتیاز و خانه های بورد مختص به او را شامل میشوند برای ذخیره اطلاعات کاربران وجود دارد. در کلاس Game الگوریتم اصلی بازی برای انجام تغییرات در بورد وجود دارد و کلاس استاتیک ConsoleColors کد تمام رنگ های قابل استفاده را نگهداری میکند.*

# الگوریتم موتور بازی

*موتور بازی نیاز به توضیح خاصی ندارد، قوانین بازی پیاده سازی شده اند.*

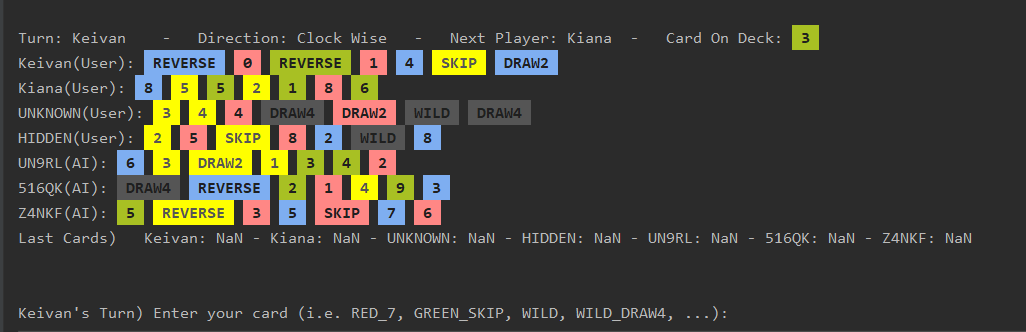
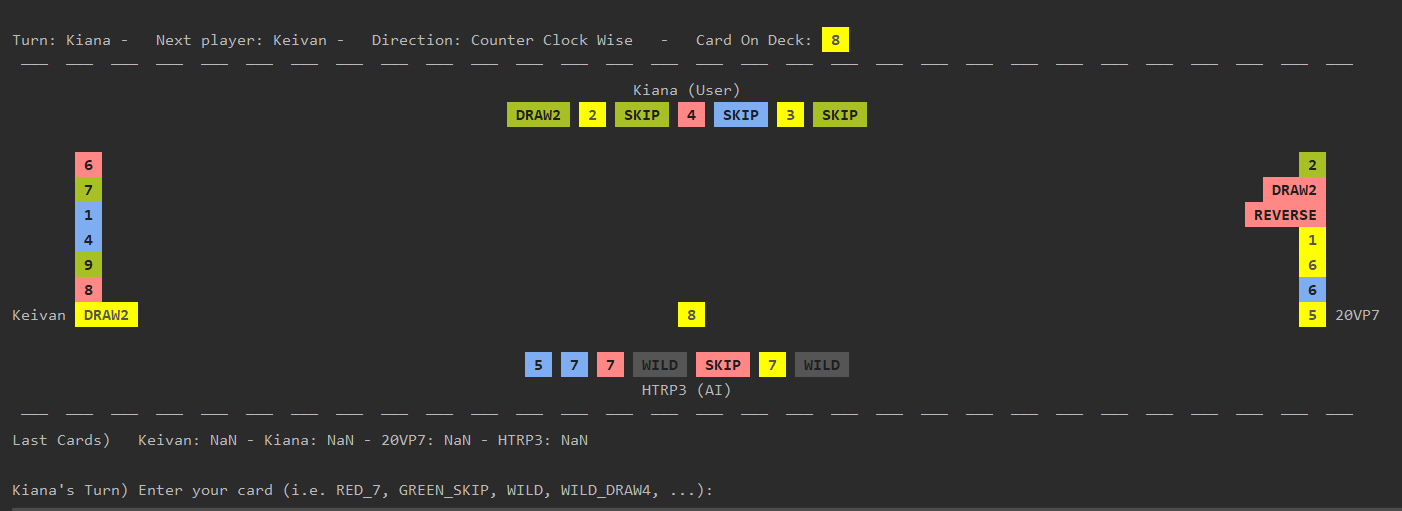
# الگوریتم هوش مصنوعی بازی

کامپیوتر یک حالت برای بازی با کاربر دارد. از کارت های خود یک کارت رندوم (و مناسب) را انتخاب و بر میگرداند.

# شیء گرا بودن و استفاده از مفهوم **Encapsulation, Encapsulation, Inheritance**

کلاس های *Player, Card, Board, Game, UNO اشیاء موجود در بازی هستند و باهم تعامل دارندهمچنین اینکه اکثر توابع و متغیر های مربوط به یک شی درون آن به صورت یک پک نوشته شده و هیچ کلاسی اطلاعی از محتویات کلاس های(اشیا) دیگر ندارد. کلاس AI توابعی برای پیش بینی حرکات بعدی خود و حرف دارد و برای این کار از برخی توابع موتور بازی استفاده میکند. نیازی به استفاده از مفهوم ارث بری نبوده و فقط کار را سخت تر میکرد! در بخش 1 و 2 پروژه ارث بری داریم.*

# نمایش 4 نفره vs 7 نفره

**

# چند نکته مهم!

* بازی دارای دو الگوریتم مختلف برای نمایش بورد است:

1. الگوریتم نمایش بورد به صورت گرافیکی: Visual Code قادر به نمایش گرافیکی نیست!!! و حتما حتما در IntelliJ تست کنید!!! (اگر در این IDE از نمایش گرافیکی استفاده کردید و چرت و پرد نشان داده شد مشکل بازی نیست 😉)؛ اگر 4 نفره بازی کنید نمایش گرافیکی بهتری خواهید گرفت و درغیر این صورت به صورت عادی
2. الگوریتم نمایش بورد به صورت ساده (کنسول): Visual Code نمایش کنسول ساده بدون رنگبندی های پیچیده تر را نشان میدهد (Visual Code PROOF!)

* بازی نمایش گرافیکی (رنگی) دارد
* تعداد پلیر نامحدود داریم
* دو نوع نمایش برد داریم
* تمام اطلاعات خواسته شده در درستور کار نمایش داده میشوند
* رنگ تصادفی به صورت خودکار انتخاب میشود
* بازی قبل 12:00 رو 27 فروردین آپلود شده.
* توجه : در هر cycle یا چرخش، بورد چرخش قبلی نمایش داده میشود و گزارش رویداد حال. یعنی گزارش حرکت بازیکن، یا ورودی یا ... یک مرحله جلو تر گزارش میشود برای راحتی انتخاب کاربر!!!! (باگ برنامه نیست!)

با تشکر – کیوان ایپچی حق - 9831073