هدف این پروژه ، پیاده سازی بازی pentago است. این پروژه حاوی 4 کلاس می باشد: کلاس Main که کلاس اصلی است، کلاس Stone که مربوط به مهره هاست، کلاس Map که مربوط به نقشه ی بازی و کلاس Player که مربوط به هر بازیکن است.

کلاس Stone :

این کلاس یک مهره ی جدید می سازد.field های این کلاس شامل نوع مهره (قرمز یا سیاه) ، شماره ی ستون ، شماره ی سطر و شماره ی بلوک مهره است که فیلد های تایپ ، شماره ی ستون و شماره ی سطر در کانستراکتور مقدار دهی شده و هم چنین با فراخوانی متد setBlockNumber که توضیح آن در ادامه آمده است ، شماره ی بلوک آن نیز تعیین می شود.

متد های این کلاس شامل:
همه ی getter و setter های مورد نیاز

متد setBlockNumber :
این متد برای تعیین شماره ی بلوک است. نحوه ی عملکرد به این گونه است که مطابق با نقشه ی بازی اگر :
شماره ی سطر و ستون < 3 باشند ----> بلوک =1
شماره ی سطر < 3 و شماره ی ستون >=3 باشد -------> بلوک =2
شماره ی سطر>= 3 و شماره ی ستون<3 باشد ---------> بلوک= 3
شماره ی سطر و ستون >=3 باشد ----------> بلوک= 4

کلاس Map :

این کلاس یک نقشه ی جدید ایجاد می کند. Field های این کلاس شامل یک آرایه ی دو بعدی از کاراکتر برای نگه داری نقشه ی اصلی بازی و یک لیست از تمامی مهره هاست. کانستراکتور این کلاس چیزی دریافت نمی کند و تنها یک آرایه ی دو بعدی کاراکتر 6 در 6 ایجاد می کند، آن را با فراخوانی setMap که توضیح آن در ادامه آمده set کرده و لیست را نیز new می کند.

متد های این کلاس شامل:
متد setMap :
این متد آرایه ی دو بعدی را مقدار دهی میکند. به این صورت که همه ی خانه ها را با '0' مقداردهی می کند.

متد updateMap :
این متد برای update کردن آرایه ی دو بعدی است. به این صورت که ابتدا با فراخوانی متد setMap همه ی خانه های آن را set کرده و سپس با پیمایش بر روی لیست مهره ها هر مهره را در جای خودش و با کاراکتر خودش قرار می دهد.
(برای مثال اگر یک مهره ی قرمز با شماره ی ستون 4 و شماره ی سطر 4 داشته باشیم ، در خانه ی 4 و 4 آرایه ی دو بعدی یک مهره ی قرمز قرار می دهد)

متد printMap :
این متد برای چاپ کردن نقشه ی بازی است. عملکرد به این صورت است که با یک پیمایش بر روی آرایه ی دو بعدی از کاراکتر یا همان map بازی ، ابتدا بلوک های 1 و 2 را چاپ می کند و سپس بلوک های 3 و 4 را.
برای زیبا سازی، خط های جداسازی با رنگ زرد و خانه های خالی با رنگ آبی (با کاراکتر '0') نمایش داده می شوند.

متد addStone :
این متد یک مهره گرفته و آن را به لیست مهره ها اضافه می کند.

متد isFull :
این متد چک میکند که نقشه پر شده است یا نه. که در صورت پر بودن ، true و در غیر این صورت false بر میگرداند.
عملکرد به این صورت است که یک پیمایش بر روی آرایه ی دو بعدی انجام می دهد و هر جایی که به خانه ای برخورد کند که در آن کاراکتری برابر '0' باشد ، true باز میگرداند. با تمام شدن پیمایش false بر میگرداند.

متد returnChar :
یک شماره ی ستون و شماره ی ردیف گرفته و کاراکتر موجود در آن نقطه در آرایه ی دو بعدی را باز می گرداند.

متد getStones :
لیست مهره ها را باز میگرداند.

کلاس Player :

این کلاس یک player جدید می سازد. field های این کلاس شامل یک لیست از مهره ها ، یک کاراکتر تایپ مهره های بازیکن(قرمز یا سیاه) و یک آبجکت از کلاس Map است که نقشه ی اصلی بازی است.در کانسراکتور باید field های map و type مقداردهی شوند و لیست نیز new می شود.

متدهای این کلاس شامل:
متد getStones :
لیست مهره های این player را باز می گرداند.

متد newStone:
یک شماره ی ستون و شماره ی ردیف از کاربر(در کنسول و با یک فاصله بین دو شماره) گرفته و در صورت معتبر بودن مکان وارد شده و در این صورت که آن مکان خالی باشد، یک stone جدید با آن شماره ی ردیف و ستون و با تایپ مهره های بازیکن ساخته و به لیست مهره های بازیکن و به لیست مهره های موجود در کلاس Map اضافه می کند.

متد rotateBlock :
یک شماره ی بلوک و یک جهت (clockwise or anti-clockwise) از کاربر (در کنسول و با یک فاصله بین شماره و جهت) گرفته و بعد از چک کردن معتبر بودن هر دو ، متد مخصوص هر بلاک که توضیحات آن در ادامه آورده شده را فراخوانی می کند. هر متد یک string جهت و یک آرایه از کل مهره های هر دو بازیکن می گیرد.(که این لیست را با استفاده از map.getStones به توابع پاس می دهیم.)

متد rotateBlock1 :
این متد برای چرخاندن بلوک اول است که یک string جهت و یک لیست از مهره های هر دو بازیکن گرفته و با یک پیمایش بر روی همه ی مهره ها، آن مهره هایی که شماره بلوکشان =1 است پیدا کرده و سپس ابتدا با فراخوانی متد displayPosition که توضیحات آن در ادامه آورده شده ، جای شماره ی سطر و ستون را با هم عوض می کند و سپس عملیات زیر را انجام می دهد:
ساعتگرد : بعد از جا به جایی x,y :

If y=0 ------> y=2
Else if y=2 -------> y=0

پادساعتگرد: بعد از جا به جایی x,y :

If x=0 -------> x=2
Else if x=2 --------> x=0

متد rotateBlock2 :

این متد برای چرخاندن بلوک دوم است که یک string جهت و یک لیست از مهره های هر دو بازیکن گرفته و با یک پیمایش بر روی همه ی مهره ها، آن مهره هایی که شماره بلوکشان =2 است پیدا کرده و سپس ابتدا با فراخوانی متد displayPosition که توضیحات آن در ادامه آورده شده ، جای شماره ی سطر و ستون را با هم عوض می کند و سپس عملیات زیر را انجام می دهد:
ساعتگرد: بعد از جا به جایی x,y :

If x=3 ------> x=0
Else if x=4 ------> x=1
Else if x=5 -------->x=2
If y=2 ------>y=3
Else if y=1 ------>y=4
Else if y=0 -------->y=5

پادساعتگرد: بعد از جا به جایی x,y :

If y=0 -----> y=3
Else if y=1 ------> y=4
Else if y=2 --------> y=5
If x=3 --------> x=2
Else if x=4 --------> x=1
Else if x=5 ---------> x=0

متد rotateBlock3 :

این متد برای چرخاندن بلوک سوم است که یک string جهت و یک لیست از مهره های هر دو بازیکن گرفته و با یک پیمایش بر روی همه ی مهره ها، آن مهره هایی که شماره بلوکشان =3 است پیدا کرده و سپس ابتدا با فراخوانی متد displayPosition که توضیحات آن در ادامه آورده شده ، جای شماره ی سطر و ستون را با هم عوض می کند و سپس عملیات زیر را انجام می دهد:
ساعتگرد: بعد از جا به جایی x,y :

If y=3 ------> y=2
Else if y=4 ------> y=1
Else if y=5 --------> y=0
If x=0 --------> x=3
Else if x=1 -----> x=4
Else if x=2 -------> x=5

پادساعتگرد: بعد از جا به جایی x,y :

If x=2 -------> x=3
Else if x=1 -------> x=4
Else if x=0 -------> x=5
If y=3 -------> y=0
Else if y=4 -------> y=1
Else if y=5 -------> y=2

متد rotateBlock4 :

این متد برای چرخاندن بلوک چهارم است که یک string جهت و یک لیست از مهره های هر دو بازیکن گرفته و با یک پیمایش بر روی همه ی مهره ها، آن مهره هایی که شماره بلوکشان =4 است پیدا کرده و سپس ابتدا با فراخوانی متد displayPosition که توضیحات آن در ادامه آورده شده ، جای شماره ی سطر و ستون را با هم عوض می کند و سپس عملیات زیر را انجام می دهد:
ساعتگرد: بعد از جا به جایی x,y :

If y=3 -----> y=5
Else if y=5 -----> y=3

پادساعتگرد: بعد از جا به جایی x,y :

If x=5 -----> x=3
Else if x=3 ------> x=5

متد displayPosition :
این متد یک مهره گرفته و جای شماره ی سطر و ستون آن را با هم عوض می کند.

متد checkStones :
این متد برای چک کردن این است که آیا 5 مهره در یک ردیف، ستون یا قطر پشت سر هم قرار گرفته اند یا نه ، که true یا false برمیگرداند.
عملکرد به این صورت است که با دو پیمایش تو در تو بر روی لیست مهره های همین بازیکن ، هر بار یک مهره را به عنوان main_stone در نظر گرفته و آن را با تک تک مهره ها مقایسه میکند به طوری که هر بار یک مهره ی دیگر را به عنوان stone در نظر میگیرد و سپس بررسی میکند که آیا main_stone و stone در یک ردیف با فاصله ی 1 ، یا در یک ستون با فاصله ی 1 و یا در یک قطر با فاصله ی رادیکال 2 قرار دارد یا نه.
اگر هر یک از این شرط ها برقرار شد، وارد آن شرط شده و یک متغیر counter=1 قرار می دهد و سپس از main_stone شروع کرده و ابتدا جلو می رود و تعداد مهره های هم رنگ (هم تایپ) را می شمارد و counter را زیاد می کند و سپس از main_stone شروع کرده و عقب می رود و تعداد مهره های هم رنگ(هم تایپ) را می شمارد و counter را زیاد می کند و سپس اگر counter >=5 بود ، true بر میگرداند. این کار را برای همه ی مهره ها امتحان کرده و در نهایت اگر هیچ جایی به counter>=5 نرسید، false برمیگرداند.

متد playerTurn :
این متد برای پیاده سازی همه ی عملکرد های لازم در هر دور از بازی بازیکن است.
ابتدا متد newStone را فراخوانی می کند و یک مهره ی جدید اضافه میکند. سپس متد های update و print را برای map صدا می زند. سپس با صدا زدن متد checkStone چک میکند که آیا بازیکن با قرار دادن این مهره ی جدید برنده شده است یا نه که اگر برنده شده دیگر کاری انجام ندهد.
اگر بازیکن هنوز برنده نشده بود، متد rotateBlock را فراخوانی میکند تا بازیکن یک بلوک را بچرخاند. در نهایت دوباره متد های update و print را برای map صدا می زند.

کلاس Main :

این کلاس ، کلاس اصلی برنامه است. در ابتدا یک آبجکت از کلاس Map ساخته می شود و متد print برای آن صدا زده می شود. سپس type مهره های هر بازیکن به صورت رندوم انتخاب میشود به این صورت که یک عدد در بازه ی (0,1] تولید شده و در صورتی که این عدد بزرگ تر از 0.5 باشد ، تایپ مهره های بازیکن اول ، قرمز و تایپ مهره های بازیکن دوم سیاه انتخاب می شود و اگر این عدد کوچکتر یا مساوی 0.5 باشد بالعکس.سپس دو آبجکت از کلاس player ساخته می شود. بعد از آن وارد یک حلقه ی بی نهایت می شویم که در ابتدای آن نوبت بازیکن اول است و متد playerTurn برای بازیکن اول صدا زده می شود.سپس

شرط به پایان رسیدن بازی با فراخوانی متد endGame که توضیح آن در ادامه آمده است، بررسی شده و و در صورتی که بازی به پایان رسیده باشد ، حلقه شکسته می شود. سپس نوبت بازیکن دوم است و متد playerTurn برای بازیکن دوم صدا زده می شود.سپس شرط به پایان رسیدن بازی با فراخوانی متد endGame بررسی شده و و در صورتی که بازی به پایان رسیده باشد ، حلقه شکسته می شود. این حلقه تا زمان خاتمه ی بازی ادامه پیدا می کند.

متد endGame:

این متد دو player و map را میگیرد و شروط خاتمه ی بازی را بررسی میکند و در صورت اتمام بازی ، true و در غیر این صورت false بر میگرداند. عملکرد به این صورت است که بررسی میکند که

اگر متد های checkStones برای هر دو بازیکن ، true برگرداند، یعنی هر دو برنده شده و بازی به پایان می رسد.

اگر متد checkStones برای یکی از بازیکن ها ، true برگرداند آن بازیکن برنده ی بازی است و بازی به پایان می رسد.

اگر متد isFull برای map ، true برگرداند یعنی نقشه ی بازی پر شده است و هیچ بازیکنی برنده نشده است و بازی به پایان می رسد.