نحوه ورودی و خروجی پروژه

چیزی که ما از شما برای ورودی و خروجی برنامه می‌خواهیم این است که تعدادی تابع که توضیح آن‌ها داده می‌شود را پیاده‌سازی کنید. نمره دهی شما صرفا از طریق این توابع خواهد بود پس در درست پیاده‌سازی کردن آن‌ها دقت کنید. اگر دوست داشتید می‌توانید یک رابط کنسولی نیز با استفاده از این توابع (یا بدون آن) برای بازی درست کنید که البته ما با آن کاری نداریم. . در هر صورت توجه داشته باشید که چیزهایی که در این فایل می‌آیند تنها مربوط به همین فاز از پروژه هستند و در فازهای بعد بازی به صورت گرافیکی خواهد بود. پس برنامه را طوی بنویسید که بخش اصلی و منطق بازی وابستگی به این توابع و رابط کاربری کنسولی (در صورت پیاده‌سازی) نداشته باشند، تا در فازهای بعد به راحتی بتوانید رابط کاربری فعلی را بردارید و یک رابط دیگر (گرافیکی) به آن اضافه کنید.

شما در کل باید چهار تابع پیاده‌سازی کنید. یک تابع برای شروع بازی که نقشه بازیکن به آن داده می‌شود و بازی با آن نقشه ساخته می‌شود. یک تابع جهت دریافت اطلاعات بازی مثل مقدار منابع و مختصات کارگرها و ... . یک تابع جهت دستور دادن به اجزای مختلف بازی و یک تابع که با صدا زدن آن بازی یک نوبت جلو می‌رود. توابع پیاده‌سازی شده بالا را در کلاس `judge.Judge` (کلاس `Judge` در پکج `judge`) قرار دهید و به همراه پروژه بفرستید. همانطور که گفته شد پیاده‌سازی این توابع الزامیست و نمره‌دهی شما فقط از طریق آن‌ها انجام می‌شود.

قبل از شروع یک نکته را برای تاکید بیشتر بگم تا اگر تا الان متوجه نشده‌اید ابهامی نباشد. بازی قرار است که به صورت نوبتی و گام به گام انجام شود. به این معنی که بازیکن به اجزای مختلف بازی یک سری دستور می‌دهد و سپس به بازی می‌گوید که یک گام جلو برود (نوبت خود را تمام می‌کند). هر بار که بازیکن نوبت خود را تمام می‌کند، اجزای بازی یک نوبت فعالیت‌های خود را جلو می‌برند. پس مثلا اگر کارگری می‌خواهد به خانه‌ای حرکت کند که با خانه‌ی فعلیش سه تا فاصله دارد، بازیکن ابتدا دستور را به کارگر می‌دهد و بعد از اینکه بازیکن سه بار نوبت خود را تمام می‌کند، کارگر به مقصد خود می‌رسد.

شناسه

هر یک از اشیا موجود در بازی (کارگرها، ساختمان‌ها و ...) باید دارای یک شناسه (`id`) منحصر به فرد باشند. این شناسه به صورت یک `String` بدون فاصله است. در طول بازی برای دادن دستورات به یک کارگر یا ساختمان از این شناسه استفاده می‌شود.

شناسه اولین کارگری که در شروع بازی در اختیار بازیکن قرار می‌گیرد باید `"derp"` باشد ولی شناسه بقیه چیزهایی که ساخته می‌شوند در اختیار خودتان است و به هر صورتی که مایلید می‌توانید بگذارید به شرطی که منحصر به فرد باشند (مثلا در حالت خیلی ساده می‌تواند صرفا یک شمارنده باشد).

در کل اشیایی که باید شناسه داشته باشند کارگرها، ساختمان‌ها، قایق‌ها و دانشجوها هستند.

توابع

در ادامه توابعی که باید پیاده‌سازی کنید توضیح داده می‌شوند

```
public static void start(char[][] map, int[][] goldMap, int[][]
stoneMap, int[][] lumberMap, int[][] foodMap);
```

اولین تابع، تابع `start` است. با صدا زده شدن این تابع بازی شروع می‌شود. ورودی‌های این تابع نقشه بازی و منابع موجود در آن را مشخص می‌کنند

ورودی `map` یک آرایه دو بعدی از کاراکتر است که نوع خانه‌های نقشه را مشخص می‌کند. خانه‌های نقشه با کاراکترهای زیر مشخص می‌شوند:

دشت	P
کوه	M
جنگل	F
آب	W

دقت کنید که در این آرایه اندیس اول شماره ردیف و اندیس دوم شماره ستون است. یعنی مثلا `map[0][3]` مربوط به ردیف اول و ستون چهارم نقشه است.

هر یک از آرگومان‌های بعدی نیز آرایه‌ی دو بعدی شامل اعداد بین `0` تا `9` است که مقدار منابع موجود در خانه‌های نقشه را مشخص می‌کند. مثلا `lumberMap[0][3]` برابر مقدار منبع چوب در ردیف اول و ستون سوم نقشه است.

واضح است که اندازه همه این آرایه‌ها با هم برابر است.

```
public static Map<String, String> info(String id);
```

این تابع برای کسب اطلاعات بازی به کار می‌رود. به این صورت که شناسه یک کارگر یا ساختمان را به عنوان ورودی می‌گیرد و مشخصات مربوط به آن را بر می‌گرداند. در صورتی هم که ورودی تابع (`id`) `null` بود، تابع باید مشخصات کلی بازی مانند مقدار منابع را برگرداند.

در ادامه مشخصاتی که برای انواع اشیا مختلف باید در خروجی تابع داده شود، ذکر شده است. نوع خروجی تابع و اینکه این مشخصات را به چه صورت باید برگردانید بعدا گفته خواهد شد.

در صورتی که ورودی تابع `null` بود مشخصات زیر باید برگردانده شوند.

Name	Value
gold	یک String شامل مقدار منبع طلای بازیکن و ظرفیت آن به صورت زیر: ظرفیت طلا / مقدار طلا مثلا: "125/200" در صورتی که شما در بازی برای منابع ظرفیت نگذاشته‌اید فقط مقدار منبع برگدانده شود، مثلا: "125"
stone	مشابه بالا برای منبع سنگ
lumber	مشابه بالا برای منبع چوب
food	مشابه بالا برای منبع غذا
knowledge	مشابه بالا برای منبع علم
buildings	یک رشته شامل شناسه‌های تمامی ساختمان‌های موجود در بازی که با فاصله از یکدیگر جدا شده اند
workers	یک رشته شامل شناسه‌های تمامی کارگرهای موجود در بازی که با فاصله از یکدیگر جدا شده اند
scholars	یک رشته شامل شناسه‌های تمامی دانشجوهای موجود در بازی که با فاصله از هم جدا شده‌اند
boats	یک رشته شامل شناسه‌های تمامی قایق‌های موجود در بازی که با فاصله از هم جدا شده‌اند

در صورتی که id داده شده به تابع مربوط به یک **کارگر** بود، باید مشخصات زیر برگردانده شوند.

Name	Value
location	مختصات کارگر در نقشه به صورت زیر: "<row> <col>" یعنی ابتدا شماره سطر کارگر در نقشه و سپس شماره ستون آن می‌آید. توجه کنید که این مختصات و تمامی مختصاتی که در ادامه می‌آیند **1 based** هستند به این معنی که از 1 شروع می‌شوند و نه 0
inventory	در صورتی که بازی شما به این صورت است که کارگران منابع را به انبار یا ساختمان اصلی حمل می‌کنند، این مشخصه نشان می‌دهد که در حال حاضر کارگر چه‌قدر از یک محصول را دارد حمل می‌کند. به صورت زیر: "[gold/stone/lumber/food] ظرفیت کارگر/مقدار حمل شده" مثلا اگر کارگر می‌تواند حداکثر 30 تا حمل کند و در حال حاضر 15 تا طلا دارد این مشخصه به صورت "gold 15/30" خواهد بود توجه کنید که کارگری که دارد در بدست آوردن یک منبع کار می‌کند باید در حین کار کردن نیز مقدار inventory او زیاد شود نه اینکه فقط موقعی که می‌خواهد آن را به انبار ببرد.
occupation	نوع فعالیتی که کارگر دارد انجام می‌دهد را مشخص می‌کند. در صورتی که کارگر دارد در یکی از منابع کار می‌کند این عبارت باید به صورت زیر باشد: "working at [gold mine/stone mine/wood camp/farm] <id>" به عنوان مثال اگر کارگر در حال کار کردن در یک معدن سنگ با شناسه "stonemine1" است، عبارت بالا به صورت زیر خواهد بود: "working at stone mine stonemine1" توجه کنید که در صورتی که کارگر در حال حمل کردن منابع به انبار یا ساختمان اصلی است نیز همان جمله باید داده شود. در صورتی که کارگر در حال ساختن یک ساختمان است "constructing <id>" داده شود که منظور از <id> شناسه ساختمان در حال ساخت است. در صورتی که کارگر دارد به سمت محل ساخت ساختمان حرکت می‌کند نیز این عبارت داده شود. در صورتی که کارگر در حال حرکت به خانه‌ای از نقشه است (به او دستور حرکت به آن خانه داده شده است) جمله زیر داده شود: "moving to <row> <col>" در صورتی هم که کارگر هیچ‌کاری نمی‌کند عبارت "idle" داده شود
buildings	یک رشته شامل شماره ساختمان‌هایی که کارگر می‌تواند بسازد[1] که با فاصله از هم جدا شده‌اند.

[1] برای اینکه احتمال خطای ناشی از اشتباه تایپ کردن اسم ساختمان‌ها و تحقیقات کم شود، انواع ساختمان‌ها و تحقیقات را با عدد نشان می‌دهیم که جدول آن در انتهای فایل آمده است.

در صورتی که id داده شده به تابع مربوط به یک **ساختمان** شود، مشخصات زیر باید برگردانده شوند.

Name	Value
building type	عدد مشخص کننده نوع ساختمان (طبق جدول آخر فایل)
blocks	مختصات خانه‌هایی که این ساختمان اشغال کرده است را مشخص می‌کند. مختصات خانه به با کاما از هم جدا شوند، به صورت زیر "<row1> <col1> , <row2> <col2> , ..." ترتیب آمدن خانه‌ها مهم نیست.

در صورتی که id داده شده به تابع مربوط به یک ساختمان از نوع **دانشگاه** باشد، علاوه بر مشخصات کلی گفته شده بزای ساختمان‌ها، مشخصات زیر نیز باید داده شوند.

Name	Value
scholars	تعداد دانشجوهای ساختمان و ظرفیت دانشگاه را به صورت زیر برمی‌گرداند: ظرفیت دانشگاه / تعداد دانشجوی‌های دانشگاه پس اگر مثلا با انجام تحقیقات ظرفیت را به 9 رسانده‌ایم و در حال حاضر 4 دانشجو در دانشگاه وجود دارد به صورت زیر خواهد بود: "4/9" در صورتی که هم ظرفیت دانشگاه نامحدود شد (با انجام تحقیقات) به جای ظرفیت دانشگاه عبارت "inf" گذاشته شود. مثلا: "4/inf"
researching	شماره تحقیق که در حال انجام است. اگر تحقیقی در حال انجام نیست عبارت "no researches in process" داده شود.
finished	شماره تحقیقات به پایان رسیده که با فاصله از هم جدا شده‌اند
available	شماره تحقیقات قابل انجام که با فاصله از هم جدا شده‌اند

در صورتی که id مربوط به یک **قایق** بود مشخصه زیر برای آن در نظر گرفته شود

Name	Value
location	مختصات کارگر در نقشه به صورت زیر: "<row> <col>" یعنی ابتدا شماره سطر کارگر در نقشه و سپس شماره ستون آن می‌آید.

در صورتی که شناسه داده شده مربوط به یکی از موارد بالا (غیر از null) بود، مشخصه زیر نیز به مشخصات اضافه شود

Name	Value
type	نوع شی را مشخص می‌کند و یکی از عبارت‌های زیر است worker building boat scholar

در صورتی که id داده شده به تابع به صورت زیر بود

"block <row> <col>"

مثلا

"block 5 8"

باید مشخصات خانه نقشه با ردیف <row> و ستون <col> داده شود. مشخصاتی که باید داده شود به صورت زیر است

Name	Value
type	اگر این خانه نقشه روشن شده است یکی از کاراکترهای P M F W که معادل با نوع خانه نقشه است. (طبق جدول صفحه 3) در صورتی که این خانه از نقشه هنوز روشن نشده است کاراکتر علامت سوال '?'
gold	در صورتی که مقدار طلای این خانه معلوم است (خانه روشن شده و تحقیق مربوطه انجام شده است) مقدار طلای درون این خانه (یک عدد بین 0 تا 9) داده شود در صورتی که معلوم نیست علامت سوال '?' داده شود.
stone	مشابه بالا برای سنگ
lumber	مشابه بالا برای چوب
food	مشابه بالا برای غذا

تابع info مشخصات خواسته شده را در قالب یک Map برمی‌گرداند. مختصری در مورد نحوه استفاده از Map در صفحه بعدی گفته شده است، در صورتی که با آن آشنایی ندارید ابتدا آن را بخوانید. (Map و به طور کلی Containerها و Genericها جز موضوعات درسی شما هستند و بعدا به صورت مفصل‌تر با آن‌ها آشنا خواهید شد ولی برای این پروژه آنچه در صفحه بعد گفته شده است کفایت می‌کند)

خروجی که تابع info می‌دهد یک Map است که keyهای آن از نوع String و شامل اسم مشخصه‌ها هستند و valueهای آن نیز از نوع String و شامل اطلاعات مربوط به آن مشخصه‌ها می‌باشند. به عنوان مثال فرض کنید که با استفاده از تابع info می‌خواهیم مختصات کارگری با شناسه "derp" را بدست بیاوریم

```
Map<String, String> derpInfo = info("derp");
String location = derpInfo.get("location");ق
```

و یا فرض کنید که می‌خواهیم تحقیق در حال انجام در دانشگاه با شناسه "uni" را بدست بیاوریم.

```
Map<String, String> uniInfo = info("uni");
String research = uniInfo.get("researching");
```

پس کاری که شما برای پیاده‌سازی تابع info باید انجام بدهید این است که ابتدا ببینید id داده شده مربوط به کی است و سپس یک Map (در واقع یک HashMap) بسازید و مشخصات آن شی که در جدول‌های بالا گفته شده را در این Map، put کنید و Map را برگردانید.

یک نکته دیگر اینکه در صورتی که id داده شده به تابع وجود نداشته باشد، تابع null برگرداند.

به صورت ساده یک Map قرار است که بین مجموعه‌ای از keyها و valueها نگاشت برقرار کند. مثلا فرض کنید که مجموعه‌ای از Stringها دارید و می‌خواهید به هر یک از آن‌ها یک عدد نسبت دهید. در این صورت به یک Map نیاز دارید که بین Stringها و عددها نگاشت برقرار کند. می‌توایند یک Map را مانند یک آرایه‌ای فرض کنید که اندیس‌های آن لزوما integer نیستند. مثلا

```
m[“hadi”] = 2.5
m[“alireza”] = 1.25
```

(در جاوا چیزی مثل بالا نداریما!! نرین اینجوری بنویسین بگین من گفتم)

در مثال بالا به "hadi" و "alireza" (اندیس‌ها) key گفته و به 2.5 و 1.25، value گفته می‌شود. در یک Map، keyهای تکراری نمی‌توانیم داشته باشیم اما value تکراری می‌توانیم.

حالا استفاده از Map در جاوا

برای تعریف یک Map به صورت زیر عمل می‌کنیم

```
Map<[key-type] , [value-type]> map;
```

که منظور از [key-type] کلاس مربوط به keyها و [value-type] کلاس مربوط به valueها است. به این صورت ما می‌توانیم objectهایی از هر کلاسی را به objectهایی از هر کلاس دیگری نگاشت کنیم. مثلا برای مثال بالا Map را به صورت زیر تعریف می‌کنیم

```
Map<String, Double> map;
```

برای اینکه در Map یک نگاشت برقرار کنیم از تابع put استفاده می‌کنیم. این تابع به ترتیب یک key و value می‌گیرد و بین آن‌ها نگاشت برقرار می‌کند.

```
map.put(“hadi”, 2.5);
map.put(“alireza”, 1.25);
```

برای اینکه value نگاشت شده به یک key را بدست بیاوریم از تابع get استفاده می‌کنیم. این تابع یک key در ورودی می‌گیرد و value مربوط به آن را برمی‌گرداند.

```
double d = map.get(“hadi”);   // d = 2.5
```

یک نکته که باید به آن توجه کنید این است که در جاوا Map یک کلاس نیست، یک interface است. (اگر نمی‌دانید interface چیست، مهم نیست. فقط در همین حد بدانید که از روی یک interface نمی‌توان instance گرفت). پس ما نمی‌توانیم بگوییم:

```
Map<String, Double> map = new Map<String, Double>();
```

در عوض باید از یکی از کلاس‌هایی که از Map مشتق می‌شوند instance بگیریم، مثل کلاس HashMap. بنابراین برای تعریف یک Map به صورت زیر عمل می‌کنیم:

```
Map<String, Double> map = new HashMap<String, Double>();
```

```
public static String action(String id, String command);
```

تابع بعدی تابع action است که برای دادن دستور به یک کارگر یا ساختمان به کار می‌رود. این تابع دو ورودی می‌گیرد، شناسه کارگر یا ساختمانی که می‌خواهیم به آن دستور بدهیم و دستور مورد نظر. خروجی این تابع نیز پیغامی است که موفقیت آمیز بودن دستور را مشخص می‌کند. در ادامه انواع دستوراتی که به کارگر و ساختمان‌های مختلف می‌توان داد آمده است.

دستوراتی که می‌توان به یک **کارگر** داد به صورت زیر است

Command	Description
move <row> <col>	کارگر به خانه با مختصات داده شده حرکت می‌کند. اگر خانه مورد نظر یکی از خانه‌های مجاور کارگر نباشد این کار بیش از یک نوبت طول می‌کشد. وقتی کارگر به مقصد رسید متوقف می‌شود. در صورتی که مختصات داده شده قابل قبول نباشد (مثلا مختضات منفی) خروجی به صورت زیر باشد: "invalid location" ممکن است که کارگر هرگز نتواند به مقصد مورد نظر برسد، (مثلا خانه مقصد کوه باشد یا اینکه خانه مقصد به دلیل ساخت ساختمان غری قابل دسترسی شود) در این صورت کارگر تا دریافت نکردن دستور جدید همچنان تلاش می‌کند به آن خانه برود و در صورتی که امکان رسیدن به آن خانه میسر شود، وارد آن می‌شود.
work <id>	کارگر مشغول به کار در ساختمانی با شناسه داده شده می‌شود. ابتدا کارگر به سمت آن ساختمان حرکت می‌کند و وقتی به آن رسید وارد آن شده و مشغول کار می‌شود. در صورتی که شناسه داده شده وجود نداشته باشد یا مربوط به ساختمانی نباشد که کارگر می‌تواند در آن کار کند خروجی تابع به صورت زیر باشد: "invalid id" در صورتی که ظرفیت محل کار گفته شده پر باشد (مثلا معدن طلا به اندازه ظرفیتش کارگر مشغول به کار داشته باشد) خروجی به صورت زیر باشد "workplace full"
build <type> <row> <col>	ساختمانی از نوع <type> در ردیف <row> و ستون <col> نقشه می‌سازد. <type> شماره ساختمان مورد نظر است. اگر کارگر در مجاورت محل ساخت قرار ندارد ابتدا باید به سمت آن حرکت کند و سپس بسازد. در صورتی نوع ساختمان داده شده صحیح نباشد خروجی زیر داده شود "invalid building" در صورتی که عدد داده شده مربوط به ساختمانی باشد که کارگر نمی‌تواند بسازد (تحقیقات کافی نبوده) خروجی تابع به صورت زیر باشد: "insufficient researches" در صورتی که منابع برای ساخت ساختمان کافی نباشد "insufficient resources" و در صورتی که مکان داده شده مجاز نباشد "invalid location"

دستوراتی که می‌توان به **ساختمان اصلی** داد به صورت زیر است

Command	Description
train	به ساختمان دستور درست کردن یک کارگر می‌دهد. در صورتی که منابع کافی نبود خروجی به صورت زیر باشد "insufficient resources" در صورتی هم که ساختمان از قبل مشغول به تربیت کارگر بوده است و هنوز تمام نشده "building is busy"

دستوراتی که می‌توان به **بندر** داد به صورت زیر است

Command	Description
train	به بندر دستور درست کردن یک قایق می‌دهد. در صورتی که منابع کافی نبود خروجی به صورت زیر باشد "insufficient resources" در صورتی هم که ساختمان از قبل مشغول به تربیت قایق بوده است و هنوز تمام نشده "building is busy"

دستوراتی که می‌توان به **دانشگاه** داد به صورت زیر است

Command	Description
train	به دانشگاه دستور تربیت یک دانشجو می‌دهد. در صورتی که منابع کافی نبود خروجی به صورت زیر باشد "insufficient resources" در صورتی هم که ظرفیت دانشگاه پر شده باشد "university full" در صورتی هم که ساختمان از قبل مشغول به تربیت دانشجو بوده است و هنوز تمام نشده "building busy"
research <number>	به دانشگاه دستور انجام تحقیق با شماره <number> می‌دهد. در صورتی که شماره تحقیق صحیح نباشد خروجی به صورت زیر باشد "invalid research" در صورتی که تحقیق مورد نظر به دلیل انجام ندادن تحقیقات پیشنیاز قابل انجام نیست خروجی به صورت زیر باشد "insufficient researches" در صورتی که منابع کافی نباشد "insufficient resources" در صورتی هم که ساختمان از قبل مشغول انجام تحقیقی بوده و هنوز تمام نشده است "building busy"

دستوراتی که می‌توان به **بازار** داد به صورت زیر است

Command	Description
exchange <amount> <from> <to>	به بازار دستور مبادله <amount> تا از کالای <from> با کالای <to> می‌دهد. مثلا فرض کنید می‌خواهیم 100 تا چوب بدهیم و در عوض سنگ بدست بیاوریم exchange 100 lumber stone در صورتی که بازیکن به اندزاه گفته شده از منبع مورد نظر ندارد (مثلا در مثال بالا 100 تا چوب ندارد) خروجی به صورت زیر باشد "insufficient resources" در صورتی هم که به دلیل پر شدن ظرفیت منابع نتوان مبادله را انجام داد (مثلا در مثال بالا در صورت انجام مبادله مدار سنگ از ظرفیت آن بیشتر شود) مهم نیست چه اتفاقی می‌افتد ولی مقدار منابع نباید از ظرفیتشان بیشتر شوند.

در صورتی هم که امکان فروختن ساختمان‌ها را گذاشته‌اید، دستور زیر برای همه ساختمان‌ها اضافه شود

Command	Description
sell	ساختمان را می‌فروشد و نصف منابع مصرف شده برای ساخت آن را به بازیکن برمی‌گرداند.

در همه موارد بالا در صورتی که دستور با موفقیت انجام شد خروجی تابع عبارت زیر باشد

```
"command successful"
```

در صورتی که شناسه داده شده به تابع وجود نداشت خروجی تابع به صورت زیر باشد

```
"invalid id"
```

در صورتی هم که شناسه وجود داشت ولی دستور داده شده معتبر نبود (چنین دستوری برای شی مورد نظر وجود نداشت) خروجی به صورت زیر باشد

```
"invalid command"
```

نکته دیگر اینکه از بین دستورات بالا دستور `exchange` بازار و دستور `sell` ساختمان‌ها در همان نوبت که داده می‌شوند اعمال می‌شوند. یعنی مثلا اگر ما یک ساختمان را `sell` بکنیم و بلافاصله بعد از آن با استفاده از تابع `info` لیست ساختمان‌ها را بدست بیاوریم، آن ساختمان دیگر نباید وجود داشته باشد. سایر دستورات تاثیر قابل مشاهده‌ای در نوبتی که دستور داده می‌شود ندارند و از نوبت‌های بعد تاثیر آن‌ها دیده می‌شود. مثلا اگر به یک کارگر دستور `move` داده شد، در آن نوبت حرکتی انجام نمی‌شود و کارگر اولین قدم را بعد از اینکه به بازی گفتیم که به نوبت بعد برود انجام می‌دهد.

علاوه بر دستورات گفته شده در بالا، تعدادی دستور دیگر هم هست که باید پیاده‌سازی کنید که مربوط به خود بازی بازیکن و روند بازی نمی‌شوند و برای تست کردن برنامه به کار می‌روند. این دستورات با همان تابع `action` داده می‌شوند و ورودی `id` برای این دستورات عبارت `"judge"` می‌باشد. پس در صورتی که تابع `action` صدا زده شد و مقدار ورودی `id` برابر با `"judge"` بود، دستورات زیر باید انجام شود. همچنین بدیهتا عبارت `"judge"` دیگر نمی‌تواند به عنوان شناسه یکی از اشیا بازی قرار بگیرد. دستورات زیر همه درجا انجام می‌شوند و نیازی به رفتن به نوبت بعد برای اعمال آن‌ها نیست

Command	Description
set <resource> <amount>	مقدار منبع <resource> را برابر با <amount> قرار می‌دهد. <resource> یکی از عبارت‌های زیر است gold stone lumber food knowledge اگر مقدار <amount> از ظرفیت بیشتر بود، مقدار منبع برابر با همان ظرفیت قرار بگیرد. خروجی خاصی نمی‌خواهد داده شود برای این دستور
worker <row> <col>	یک کارگر در مختصات داده شده قرار می‌دهد. این کارگر درجا ساخته شده و آنجا قرار می‌گیرد و کاری با ساختمان اصلی و زمان ساخت ندارد. همچنین از منابع کم نمی‌شود. در صورتی که کارگر ساخته شد خروجی تابع برابر با شناسه کارگر ساخته شده باشد. در صورتی هم که به هر دلیل ساخته نشد (مثلا مختصات صحیح نباشد) خروجی null باشد.
move <id> <row> <col>	کارگر با شناسه <id> را به خانه با مختصات داده شده منتقل می‌کند. کارگر بلافاصله در همان نوبت به آن خانه می‌رود. خروجی خاصی نمی‌خواهد داده شود.
building <type> <row> <col>	یک ساختمان از نوع <type> در مختصات داده شده قرار می‌دهد. مانند بالا، ساختمان درجا ساخته می‌شود و زمان ساخت ندارد. همچنین از منابع کاسته نمی‌شود. در صورتی که ساختمان ساخته شد شناسه آن به عنوان خروجی داده شود و در غیر این صورت null داده شود.
scholar	یک دانشجو در دانشگاه درست می‌کند و شناسه آن را برمی‌گرداند. دانشجو درجا ساخته می‌شود و منابع کم نمی‌شود. در صورتی که ظرفیت دانشگاه پر شده باشد تابع null برگرداند
research <number>	تحقیق با شماره <number> را انجام می‌دهد. این تحقیق درجا انجام می‌شود و طول نمی‌کشد و منابع مصرف نمی‌شود. همچنین نیازی به رعایت پیشنیازی تحقیقات نیست و حتی اگر تحقیقات قبلی انجام نشده باید بتوان آن را انجام داد. شاید سوال براتون پیش بیاید که در این صورت اگر مثلا تحقیقی که پیشنیازی آن رعایت نشده انجام گیرد، تحقیقات بعدی باید باز شوند یا نه، یا سوالات شبیه به این. کلا مهم نیست چه اتفاقی بیفتد. این دستور صرفا برای تست درست بودن تاثیرات تحقیق‌ها در بازی است. یعنی با وارد شدن این دستور شما کافی است فقط تاثیر تحقیق مورد نظر را اعمال کنید (مثلا مقدار تولید طلا افزایش پیدا کند و ...) خروجی خاصی نمی‌خواهد داده شود برای این دستور
explore <row> <col>	خانه با مختصات داده شده از نقشه را روشن می‌کند (مانند اینکه کارگری به آن سر زده باشد)
explore all	تمام نقشه را روشن می‌کند

```
public static String[] nextTurn();
```

تابع آخر برای تمام کردن یک نوبت از بازی است. این تابع ورودی ندارد و یک آرایه از String برمی‌گرداند. این آرایه شامل پیغام‌هایی از وقایعی است که در ابتدای نوبت بعد در بازی اتفاق می‌افتد.

پیغام‌هایی که باید در این آرایه وجود داشته باشند به صورت زیر است.

در صورتی که در این نوبت تربیت یک کارگر، دانشجو و یا قایق به پایان رسیده باشد به ترتیب پیغام‌های زیر باید داده شود.

```
worker trained <id>
scholar trained <id>
boat trained <id>
```

که در اینجا به جای <id> شناسه کارگر، دانشجو و یا قایق جدیدی که درست شده است باید قرار بگیرد.

در صورتی که یک ساختمان شروع به ساختن شد پیغام زیر داده شود

```
construction started <id>
```

این پیغام وقتی داده شود که کارگر به محل مورد نظر رفته و ساختن ساختمان واقعا شروع شده باشد و فضای مورد نظر از نقشه اشغال شده باشد. منظور از <id> شناسه ساختمان جدید در حال ساخت است.

وقتی که ساخت یک ساختمان به پایان رسید پیغام زیر داده شود

```
construction finished <id>
```

که <id> شناسه آن ساختمان است.

در صورتی که یک تحقیق در دانشگاه به پایان رسید پیغام زیر داده شود

```
research finished <number>
```

که <number> شماره تحقیق به پایان رسیده است.

در صورتی که یک کارگر یا دانشجو به دلیل کبمود غذا مرد پیغام‌های زیر داده شوند

```
worker died <id>
scholar died <id>
```

یک مثال. فرض کنید می‌خواهیم یک کارگر بسازیم. از طریق تابع `action` به ساختمان اصلی دستور ساخت کارگر را می‌دهیم. از آنجا که ساخت کارگر دو نوبت طول می‌کشد، دفعه اولی که `nextTurn` را صدا می‌زنیم اتفاقی نمی‌افتد ولی دفعه دوم پیغام ساخته شدن کارگر در خروجی تابع داده می‌شود.

یک نکته در مورد ساختمان‌های در حال ساخت. با ساختمانی که در حال ساخت است باید مثل یک ساختمانی که هیچ ویژگی خاصی ندارد برخورد کنید. مثلا اگر یک دانشگاه در حال ساخته شدن بود و تابع `info` روی آن صدا زده شد، مشخصه‌های **"building type"** و **"blocks"** باید داده شوند ولی تا وقتی ساختمان کامل نشده است مشخصات **"scholars"** و ... نباید داده شوند. یا مثلا اگر یک معدن در حال ساخته شدن است و به کارگر دستور کارکردن در آن داده شود، باید پیغام **"invalid id"** برگردانده شود.

برای تاکید بیشتر دوباره می‌گم که تمامی جاهایی که از توابع گفته شده مختصاتی گرفته می‌شود یا داده می‌شود، شماره سطر و ستون‌ها از **1** شروع می‌شوند و نه از **0**.

جدول شماره تحقیقات

Name	Number	Name	Number
Resources	1	Tax	14
Agriculture	2	Project Management	15
Mining	3	Market	16
Lumber Mill	4	Team Work	17
Ranching	5	Strategic Proj. Man.	18
Irrigation	6	Science	19
Refinery	7	Alphabet	20
Carpentry	8	Mathematics	21
Fishing	9	Engineering	22
Economy	10	School	23
Micro Econ.	11	University	24
Macro Econ.	12	CERN	25
Parsimony	13		

جدول شماره ساختمان‌ها

Name	Number
Main Building	1
University	2
Gold Mine	3
Stone Mine	4
Farm	5
Wood Camp	6
Stockpile	7
Market	8
Port	9