

رسیدن اتوبوس

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

برای مسابقات حضوری پردیس امسال، n اتوبوس در نظر گرفته شده است. این اتوبوس‌ها با اعداد ۱ تا n شماره‌گذاری شده‌اند و همگی در میدان آزادی قرار دارند و قصد دارند به محل مسابقات برسند.

اتوبوس شماره i مسیری با طول x_i کیلومتر را طی می‌کند و سرعت ثابتی برابر با v_i کیلومتر بر ساعت دارد.

کبری می‌خواهد در سریع‌ترین زمان ممکن خود را به مسابقه امروز برساند. به او کمک کنید تا کمترین زمان ممکن برای رسیدن به مقصد را محاسبه کند.

ورودی

در سطر اول ورودی، یک عدد صحیح و مثبت n داده می‌شود که نشان‌دهنده تعداد اتوبوس‌ها است.

$$1 \leq n \leq 1000$$

در n سطر بعدی، در هر سطر دو عدد صحیح x_i و v_i داده می‌شود که به ترتیب فاصله و سرعت اتوبوس شماره i را نشان می‌دهند.

$$1 \leq x_i \leq 2000, \quad 1 \leq v_i \leq 200$$

خروجی

در تنها سطر خروجی، کمترین زمان ممکن برای رسیدن به محل برگزاری مسابقات را محاسبه کنید. انتظار می‌رود مقدار خروجی شما با پاسخ واقعی کمتر از 10^{-6} اختلاف داشته باشد.

مثال‌ها

ورودی نمونه ۱

4
10 20
100 20
5 30
30 5

خروجی نمونه ۱

0.166666666667

چهار اتوبوس برای رسیدن به محل مسابقات از میدان آزادی به محل برگزاری مسابقات در نظر گرفته شده است.

- اتوبوس اول از راهی می‌رود که 10 کیلومتر را طی می‌کند و سرعت آن 20 کیلومتر بر ساعت است. پس بعد از 0.5 ساعت می‌رسد.
- اتوبوس دوم از راهی می‌رود که 100 کیلومتر را طی می‌کند و سرعت آن 20 کیلومتر بر ساعت است. پس بعد از 5 ساعت می‌رسد.
- اتوبوس سوم از راهی می‌رود که 5 کیلومتر را طی می‌کند و سرعت آن 30 کیلومتر بر ساعت است. پس بعد از 0.166... ساعت می‌رسد.
- اتوبوس چهارم از راهی می‌رود که 30 کیلومتر را طی می‌کند و سرعت آن 5 کیلومتر بر ساعت است. پس بعد از 6 ساعت می‌رسد.

ورودی نمونه ۲

2
100 150
1000 20

خروجی نمونه ۲

0.666666666667

دو اتوبوس برای رسیدن به محل مسابقات از میدان آزادی به محل برگزاری مسابقات در نظر گرفته شده است.

- اتوبوس اول از راهی می‌رود که 100 کیلومتر را طی می‌کند و سرعت آن 150 کیلومتر بر ساعت است. پس بعد از 0.666... ساعت می‌رسد.
- اتوبوس دوم از راهی می‌رود که 1000 کیلومتر را طی می‌کند و سرعت آن 20 کیلومتر بر ساعت است. پس بعد از 50 ساعت می‌رسد.

فلای‌برد بی‌کیفیت

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

احتمالاً بازی *Flappy Bird* را دیده‌اید. در این بازی، یک پرنده از سمت چپ به راست حرکت می‌کند. مانع‌ها در این بازی یا بخشی از بالا یا پایین صفحه را مسدود کرده‌اند و هدف این است که پرنده بدون برخورد به موانع به انتهای بازی برسد.



در نسخه‌ی ساده‌شده‌ی این بازی، فرض کنید کل محیط بازی به صورت یک جدول $2 \times n$ است. هر خانه از جدول می‌تواند یکی از سه وضعیت زیر را داشته باشد:

- در آن خانه پرنده قرار دارد (دقیقاً یک خانه شامل پرنده است).
- در آن خانه مانع وجود دارد.
- آن خانه خالی است.

در ابتدا، می‌دانیم که پرنده در خانه‌ی بالا-چپ قرار دارد و قصد دارد به خانه‌ی پایین-راست برود (تضمین می‌شود که این دو خانه هیچگاه مانع ندارند). پرنده می‌تواند در هر حرکت به خانه‌های **مجاور ضلعی** (در صورت نبود مانع) حرکت کند. سوال این است که آیا پرنده می‌تواند مسیری برای رسیدن به مقصد پیدا کند یا نه؟

برای درک بهتر سوال، به نمونه‌ها مراجعه کنید.

ورودی

در سطر اول ورودی، عدد صحیح و مثبت n داده می‌شود که تعداد ستون‌های جدول را نشان می‌دهد.

$$1 \leq n \leq 100$$

در دو سطر بعدی، هر سطر شامل n کاراکتر است که کاراکتر سطر i ام و ستون j ام وضعیت خانه‌ی متناظر در جدول را مشخص می‌کند. کاراکتر X نشان‌دهنده‌ی مانع و کاراکتر 0 نشان‌دهنده‌ی باز بودن خانه است.

خانه‌ی سطر اول از ستون اول و خانه‌ی سطر دوم از ستون n ام مبدا و مقصد پرنده هستند و همیشه با 0 مشخص می‌شوند.

خروجی

در تنها سطر خروجی، در صورت وجود مسیر، رشته‌ی `((:!Hooraaaay` و در غیر این صورت رشته‌ی `Awww:` را چاپ کنید.

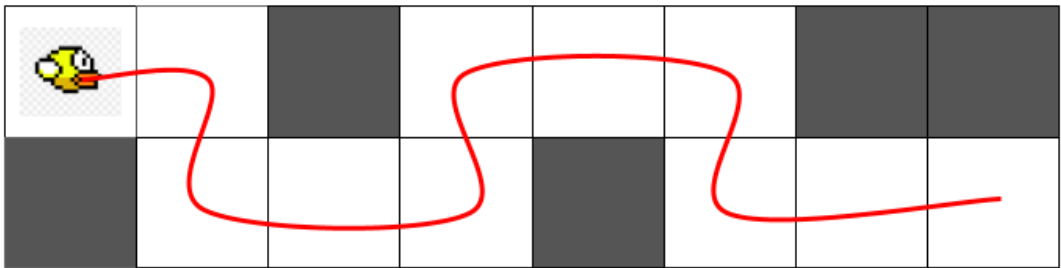
مثال‌ها

ورودی نمونه ۱

```
8
00X000XX
X000X000
```

خروجی نمونه ۱

Hooraaay! :))

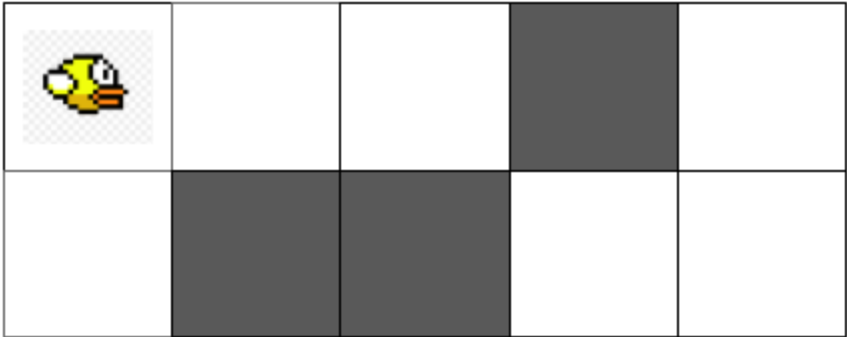


ورودی نمونه ۲

5
000X0
0XX00

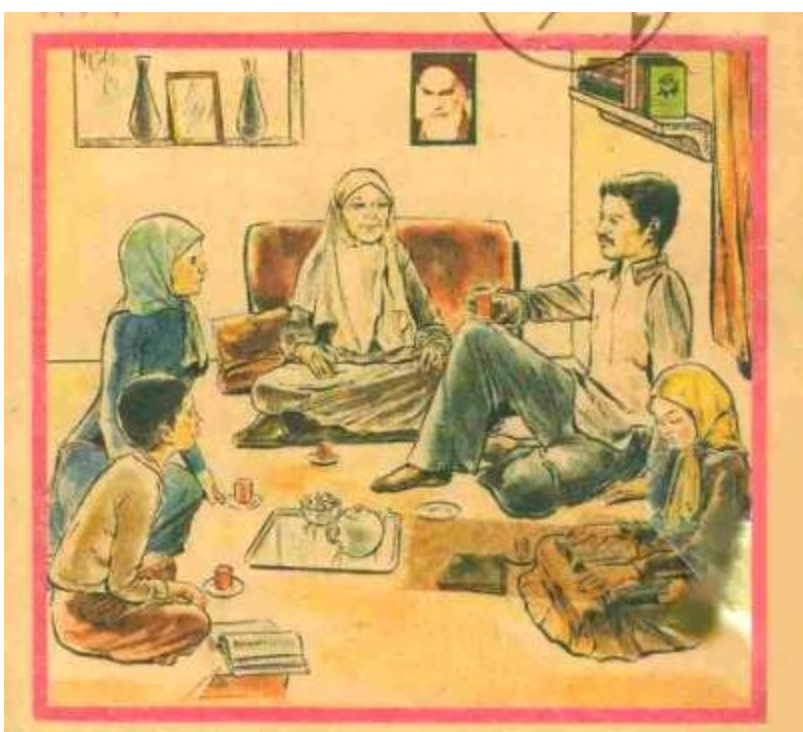
خروجی نمونه ۲

Awww:((



هایلایتر زبان هاشمی

مسئولین پارک علم و فناوری پردیس تصمیم دارند تا در سری رویداد بعدی *المپیک فناوری*، یک مسیر راه جدید برای زبان‌های برنامه‌نویسی **بومی** به ویژه زبان برنامه نویسی **آقای هاشمی** برگزار کنند. **نیما** که مسئول طراحی محتوای این مسیر راه جدید و ویژه و تولید داخلی است، به دلیل کمتر شناخته شده بودن **زبان برنامه نویسی آقای هاشمی**، دسترسی به یک **سینتکس هایلایتر (Syntax Highlighter)** مناسب برای ژینگول و خوشگل کردن کدهای رویداد را ندارد و از شما برای پیاده سازی این کد هایلایتر کمک می‌خواهد!



جزئیات پروژه

پروژه‌ی اولیه را از این لینک دانلود کنید.

▼ ساختار فایل‌ها

```
1 | src
2 | └─ CommonMark
3 |   └─ CommonMarkConverter.php
4 |
```



```

5 |   └─ Parser.php
6 | └─ Languages
7 |   └─ Hashemi
8 |     └─ HashemiLanguage.php
9 |     └─ Injections
10 |        └─ JsonInjection.php
11 |        └─ Patterns
12 |           └─ HmBuiltinPattern.php
13 |           └─ HmDoubleQuoteValuePattern.php
14 |           └─ HmFunctionCallPattern.php
15 |           └─ HmKeywordPattern.php
16 |           └─ HmMultilineCommentPattern.php
17 |           └─ HmNumberPattern.php
18 |           └─ HmOperatorPattern.php
19 |           └─ HmSinglelineCommentPattern.php
20 |           └─ HmTripleDoubleQuoteStringPattern.php
21 | └─ Themes
22 |   └─ DefaultTheme.php
23 |   └─ ThemeManager.php
24 | └─ Tokens
25 |   └─ GroupTokens.php
26 |   └─ TokenAnalyzer.php
    └─ TokenValidator.php

```

▼ راه اندازی پروژه

برای اجرای پروژه، باید **php** و **composer** را از قبل نصب کرده باشید.

- پروژه‌ی اولیه را دانلود و از حالت فشرده خارج کنید.
- دستور `composer install` را در پوشه‌ی اصلی پروژه برای نصب نیازمندی‌ها اجرا کنید.
- دستور `php -S localhost:8000 -t tests/` را در پوشه‌ی اصلی پروژه اجرا کنید. پروژه از طریق آدرس <http://localhost:8000> در دسترس خواهد بود.

تست پروژه

برای **تست پروژه** لازم است تا به مسیر `/tests` در مرورگر خود بروید. این صفحه یک فایل کد هاشمی را که از قبل در پروژه برای تست شما قرار داده شده **هایلایت کرده** و به شما **نمایش** می‌دهد.

- شما می‌توانید با **مقایسه** کد هایلایت شده در صفحه‌ی لود شده با کدهای هایلایت شده در تصاویر استفاده شده در سوال از درستی کارکرد کد خود **اطمینان** حاصل کنید.
- تست کد با استفاده از مقایسه خروجی کد هایلایت شده توسط پروژه آپلود شده **توسط شما** و کد هایلایت شده **توسط داوری خودکار** سوال مورد ارزیابی قرار خواهد گرفت.

▼ لینک‌های مفید

در بخش زیر لیستی از **لینک‌های مفید** وبسایت‌ها و مستندات که می‌تواند به شما در پیاده‌سازی این سوال کمک کند برای شما قرار گرفته است:

- وبسایت *Regexr*
- وبسایت *Regex101*
- فایل pdf برگه تقلب *Regex*
- مستندات زبان برنامه‌نویسی هاشمی
- لینک پروژه زبان برنامه‌نویسی هاشمی
- لینک پروژه اصلی *tempestphp/highlight* (بعد از کلیک روی لینک از اتصال از شبکه را قطع کنید).

پروژه اصلی `tempestphp/highlight`

در این سوال از نسخه 2.10.2 پروژه آماده کد هایلایتر `tempestphp/highlight` استفاده کرده و با توسعه‌ی آن، **زبان برنامه نویسی آقای هاشمی** را نیز به آن اضافه خواهیم کرد. شما در این سوال قرار است تا با توجه به توضیحات ساختارهای گفته شده، با استفاده از **عبارات باقاعده** (*Regular expressions*) نواحی و اجزای مختلف یک کد را **تشخیص داده** و به هسته‌ی اصلی این کد هایلایتر بگویید تا با **توکن تایپ** (*TokenType*) مناسب آن را برای شما **هایلایت** کند.

▼ **بررسی بیشتر ساختار کد هایلایتر `tempestphp/highlight` (مهم، حتما مطالعه شود!)**

بررسی بیشتر ساختار نسخه 2.10.2 کد هایلایتر tempestphp/highlight

ساختار این کد هایلایتر به این شکل است که سه مفهوم اصلی به صورت *languages*, *injections* و *patterns* را از پیش پیاده‌سازی کرده است. در این بخش به بررسی این بخش‌ها می‌پردازیم و سپس به سراغ مواردی می‌رویم که از شما انتظار داریم تا در این سوال پیاده‌سازی کنید:

۱. ساختار **الگوها** (*Patterns*): یک **الگو** (*Pattern*) نمایانگر قسمتی از کد است که باید **هایلایت** (*Highlight*) شود. یک الگو می‌تواند هدفش یک کلمه کلیدی مانند `return` یا `class` باشد یا می‌تواند هر قسمتی از کد باشد، مانند یک کامنت: `/* this is a comment */` یا حتی فراخوانی یک تابع. هر الگو با یک کلاس ساده نمایش داده می‌شود که یک **عبارت باقاعده** (*Regular expression*) و یک `TokenType` دارد. عبارت `Regex` برای تطبیق محتوای مرتبط با این الگوی خاص استفاده می‌شود، در حالی که `TokenType` یک مقدار از نوع `enum` است که تعیین می‌کند چگونه آن الگوی خاص هایلایت شود. برای بررسی مثال‌های بیشتر از **این لینک** اقدام کنید.

توجه داشته باشید که در این سوال شما صرفاً قرار است تا از `TokenType` های پیشفرض استفاده کنید و نیازی به پیاده‌سازی جدیدی برای این مورد نیست. در هر بخش `TokenType` مربوطه به شما داده می‌شود تا آن را در ساختار کد قرار دهید.

۲. ساختار **تزریق‌ها** (*Injections*): ساختار بعدی مربوط به بخش **تزریق‌ها** (*Injections*) می‌باشد. از تزریق‌ها برای هایلایت کردن زبان‌های مختلف در یک بلوک کد استفاده می‌شوند. برای مثال: `HTML` می‌تواند شامل `CSS` باشد که باید به شیوه درست در کنار `HTML` هایلایت شود. برای بررسی مثال‌های بیشتر از **این لینک** اقدام کنید. توجه داشته باشید که در این سوال شما صرفاً قرار است تا در قالب `z farzand` در کدهای زبان هاشمی خود از ساختار `Json` استفاده کنید.

۳. ساختار **زبان‌ها** (*Languages*): آخرین ساختار در این کد هایلایتر، ساختار **زبان‌ها** (*Languages*) می‌باشد که همان نماینده زبان‌های برنامه‌نویسی است به صورت کلاس‌هایی همراه با پیاده‌سازی **الگوها** (*Patterns*) و **تزریق‌ها** (*Injections*) می‌باشد. برای بررسی مثال‌های بیشتر از **این لینک** اقدام

کنید. زبانی که شما در این سوال به سراغ آن خواهید رفت زبان برنامه‌نویسی آقای هاشمی (HashemiLanguage) خواهد بود.

▼ بررسی بیشتر یک مثال مهم در پیاده سازی الگوها (Patterns) (مهم، حتما مطالعه شود!)

در این بخش به بررسی یک مثال مهم از شیوه تشخیص و هایلایت کردن الگوها (Patterns) مشخص شده در هر کدام از بخش‌های یک زبان می‌پردازیم. به مثال زیر از پیاده‌سازی هایلایت کردن namespace ها در php دقت کنید:

```
1 use Tempest\Highlight\IsPattern;
2 use Tempest\Highlight\Pattern;
3 use Tempest\Highlight\Tokens\TokenType;
4
5 final readonly class NamespacePattern implements Pattern
6 {
7     use IsPattern;
8
9     public function getPattern(): string
10    {
11        return 'namespace (<match>[\w\\\\]+)';
12    }
13
14    public function getTokenType(): TokenType
15    {
16        return TokenType::TYPE;
17    }
18 }
```

- در مثال بالا یک الگوی جدید با نام NamespacePattern ایجاد شده است که مسئول هایلایت کردن namespace ها می‌باشد. همانطور که مشاهده می‌کنید در تابع getPattern یک عبارت باقاعده return شده است که نمایانگر انتخاب‌گر آن ساختار خاص (در اینجا namespace ها) در کد می‌باشد. همچنین در تابع getTokenType یک Enum از نوع TokenType برگردانده می‌شود. در هر بخش مقدار TokenType ای که باید در این ساختار return کنید به شما معرفی می‌شود.

- این Regex ها باید شامل یک گروه با نام `match` باشند که به این صورت نوشته می‌شود: (`<match>...`). این گروه نمایانگر بخشی از کد است که در واقع هایلایت خواهد شد.

توجه داشته باشید که هر الگو باید یک **گروه تطبیق** (*Capture Group*) نام‌گذاری شده در Regex داشته باشد که نام آن `"match"` باشد. هسته هایلایتر محتوایی که درون این گروه تطبیق پیدا کرده است را هایلایت خواهد کرد.

برای مثال، این regex `(?<match>[\w\\\/\s]+)` namespace می‌گوید که هر خطی که با کلمه namespace شروع شود باید در نظر گرفته شود، اما **تنها** بخشی که درون گروه نام‌گذاری شده (`<match>...`) قرار دارد هایلایت می‌شود اما خود واژه namespace چون در گروه match در نظر گرفته نشده است هایلایت **نخواهد** شد.

افزودن زبان برنامه نویسی جدید به `tempestphp/highlight`

شما باید پوشه Hashemi که کدهای مربوط به **هایلایتر زبان هاشمی** در آن قرار دارد را مطابق با **مستندات این زبان** و توضیحات گفته شده در بخش زیر کامل و در پایان این بخش آن را آپلود کنید. **توجه کنید** که **هایلایتر شما** حتما باید در **ساختار زیر عمل کند** تا **نمره کامل را دریافت کند**. همچنین شما تنها مجاز به ویرایش فایل‌های موجود هستید و نمی‌توانید فایل‌های جدیدی را به این بخش اضافه کنید.

▼ ساختار کامنت‌های تک‌خطی در زبان هاشمی و پیاده‌سازی کلاس `HmSinglelineCommentPattern`

ساختار **کامنت‌های تک‌خطی** در زبان برنامه‌نویسی آقای هاشمی مطابق با **مستندات این زبان**، مانند بسیاری از زبان‌های دیگر به صورت زیر می‌باشد:

```
1 | // yek comment
```

شما باید کلاس `HmSinglelineCommentPattern` را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند کامنت‌های تک‌خطی را با استفاده از `TokenType` مربوط به کامنت یعنی `COMMENT` رنگی کند.

 `HmSinglelineCommentPattern.php`

```

1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmSinglelineCommentPattern implements Pattern
12 {
13     use IsPattern;
14
15     public function getPattern(): string
16     {
17         return ""; // TODO: Implement
18     }
19
20     public function getTokenType(): TokenTypeEnum
21     {
22         return TokenTypeEnum::TYPE; // TODO: Implement
23     }
24 }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به کامنت‌های تک‌خطی را به درستی هایلایت کند.

▼ ساختار کامنت‌های چندخطی در زبان هاشمی و پیاده‌سازی کلاس HmMultilineCommentPattern
 ساختار کامنت‌های چندخطی در زبان برنامه‌نویسی آقای هاشمی مطابق با *مستندات این زبان*، مانند بسیاری از زبان‌های دیگر به صورت زیر می‌باشد:

```

1  /*
2  in teke az code pardazesh nakhahad shod
3  */
4


```

```

5
6 /**
7  * baraye neveshtan e mostanadat
   */

```

شما باید کلاس `HmMultilineCommentPattern` را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند کامنت‌های چندخطی را با استفاده از `TokenType` مربوط به کامنت یعنی `COMMENT` رنگی کند.

 `HmMultilineCommentPattern.php`

```

1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11  final readonly class HmMultilineCommentPattern implements Pattern
12  {
13      use IsPattern;
14
15      public function getPattern(): string
16      {
17          return ""; // TODO: Implement
18      }
19
20      public function getTokenType(): TokenTypeEnum
21      {
22          return TokenTypeEnum::TYPE; // TODO: Implement
23      }
24  }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به کامنت‌های چندخطی را به درستی هایلایت کند.

▼ ساختار کلمات کلیدی در زبان هاشمی و پیاده‌سازی کلاس HmKeywordPattern

کلمات کلیدی در هر زبان برنامه‌نویسی، کلماتی هستند که توسط این زبان‌ها برای اعمال مختلف رزرو شده‌اند. در زبان برنامه‌نویسی هاشمی، کلمات `ta` ، `bede` ، `bebin` ، `age` ، `bood` ، `na?` ، `zirsakht` جزو کلمات کلیدی به حساب می‌آیند.

شما باید کلاس `HmKeywordPattern` را به صورت زیر در فایل با همین نام پیاده‌سازی کنید تا کد هایلایت بتواند کلمات کلیدی را با استفاده از `TokenType` مربوط به آن یعنی `KEYWORD` رنگی کند.

 HmKeywordPattern.php

```
1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmKeywordPattern implements Pattern
12 {
13     use IsPattern;
14
15     public function getPattern(): string
16     {
17         return ""; // TODO: Implement
18     }
19
20     public function getTokenType(): TokenTypeEnum
21     {
22         return TokenTypeEnum::TYPE; // TODO: Implement
23 }
```



```
23 | }  
24 | }
```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به کلمات کلیدی را به درستی هایلایت کند.

▼ ساختار توابع پیش‌ساخته در زبان هاشمی و پیاده‌سازی کلاس HmBuiltinPattern

توابع پیش‌ساخته در زبان برنامه‌نویسی آقای هاشمی مانند سایر زبان‌های برنامه‌نویسی، مسئولیت انجام برخی موارد حیاتی از جمله چاپ یا پردازش برخی رشته‌های متنی را دارند. در زبان هاشمی توابع پیش‌ساخته به صورت کلمات `azinja`، `bechap` و `jfarzand` می‌باشند.

شما باید کلاس `HmBuiltinPattern` را به صورت زیر در فایل با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند توابع پیش‌ساخته را با استفاده از `TokenType` مربوط به آن یعنی `TYPE` رنگی کند.

 HmBuiltinPattern.php

```
1  <?php  
2  
3  declare(strict_types=1);  
4  
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;  
6  
7  use Tempest\Highlight\IsPattern;  
8  use Tempest\Highlight\Pattern;  
9  use Tempest\Highlight\Tokens\TokenTypeEnum;  
10  
11  final readonly class HmBuiltinPattern implements Pattern  
12  {  
13      use IsPattern;  
14  
15      public function getPattern(): string  
16      {  
17          return ""; // TODO: Implement  
18      }  
19  }
```

```

20 |     public function getTokenType(): TokenTypeEnum
21 |     {
22 |         return TokenTypeEnum::TYPE; // TODO: Implement
23 |     }
24 | }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به توابع پیش‌ساخته را به درستی هایلایت کند.

▼ ساختار فراخوانی و تعریف توابع در زبان هاشمی و پیاده‌سازی کلاس HmFunctionCallPattern

فراخوانی‌ها و تعریف توابع در زبان برنامه نویسی هاشمی به شکل زیر می‌باشد:

```

1 | bebin jam(alef, be) {
2 |     javab = alef + be;
3 |     bede javab;
4 | }
5 |
6 | bebin azinja() {
7 |     ye_adad = 100;
8 |     ye_adad_dige = 200;
9 |     natiye = jam(ye_adad, ye_adad_dige);
10 |    bechap(natiye);
11 | }

```

- منظور از هایلایت کردن تعاریف توابع در مثال بالا، هایلایت کردن عبارت `jam` در خط اول (هنگام تعریف تابع) و در خط هشتم (در فراخوانی تابع) می‌باشد.

شما باید کلاس `HmFunctionCallPattern` را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایت بتواند تعریف و فراخوانی توابع را با استفاده از `TokenType` مربوط به آن یعنی `PROPERTY` رنگی کند.

```

1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmFunctionCallPattern implements Pattern
12 {
13     use IsPattern;
14
15     public function getPattern(): string
16     {
17         return ""; // TODO: Implement
18     }
19
20     public function getTokenType(): TokenTypeEnum
21     {
22         return TokenTypeEnum::TYPE; // TODO: Implement
23     }
24 }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به فراخوانی و تعریف توابع را به درستی هایلایت کند.

▼ ساختار استفاده از اعداد در زبان هاشمی و پیاده‌سازی کلاس HmNumberPattern

ساختار دیگری که شما باید در این هایلایتر پیاده‌سازی کنید مربوط به بخش استفاده از اعداد در این زبان می‌باشد. اعداد می‌توانند در بخش‌های مختلف کد از جمله در شرط‌ها، در مقداردهی متغیرها و در فراخوانی توابع استفاده شوند. در زبان هاشمی تنها از اعداد صحیح، اعداد اعشاری و نمایش اعداد به صورت نمادگذاری علمی پشتیبانی می‌شود.


```

1  bebin factorial(n) {
2      age (n < 1) bood {
3          bede 1;
4      }
5      bede factorial(n-1) * n;
6  }
7
8  bebin tarkib(k, n) {
9      soorat = factorial(n);
10     makhraj = factorial(k) * factorial(n-k);
11     javab = soorat / makhraj;
12     bede javab;
13 }
14
15 bebin azinja() {
16     f = tarkib(5, 2);
17     bechap(f);
18 }

```

- مثالی از استفاده از اعداد در فراخوانی توابع، شرط ها و ... در زبان برنامه نویسی هاشمی

شما باید کلاس HmNumberPattern را به صورت زیر در فایلی با همین نام پیاده سازی کنید تا کد هایلایتر بتواند تعریف و فراخوانی توابع را با استفاده از TokenType مربوط به آن یعنی NUMBER رنگی کند.

 HmNumberPattern.php

```

1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmNumberPattern implements Pattern
12

```

```

13 {
14     use IsPattern;
15
16     public function getPattern(): string
17     {
18         return ""; // TODO: Implement
19     }
20
21     public function getTokenType(): TokenTypeEnum
22     {
23         return TokenTypeEnum::TYPE; // TODO: Implement
24     }
25 }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به استفاده از اعداد را به درستی هایلایت کند.

▼ ساختار عملگرها در زبان هاشمی و پیاده‌سازی کلاس HmOperatorPattern

عملگرها در زبان آقای هاشمی در سه دسته‌بندی به شکل عملگرهای عددی `+-%/*` ، عملگرهای منطقی به صورت‌های `==` ، `<=` ، `>=` ، `!=` ، `<` ، `>` و عملگر انتساب `=` باشد. توجه کنید که عملگرهای دو کاراکتری، مثلاً `==` باید به صورت یکجا به عنوان یک عملگر در نظر گرفته شود و نه به صورت دو عملگر `=` و `=` .

```

1  age (1==1) bood {
2      bechap("doroste");
3  } na? {
4      bechap("dorost nist, ye fekri barash bokon");
5  }
6
7  adadeMan = 0;
8  majmoo = 0;
9
10 ta (adadeMan<11) bood {
11     majmoo = majmoo + adadeMan;
12     adadeMan = adadeMan + 1;
13 }

```

- مثالی از انواع استفاده‌ها از عملگرها در بخش‌های مختلف زبان هاشمی

شما باید کلاس `HmOperatorPattern` را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند عملگرهای مختلف را با استفاده از `TokenType` مربوط به آن یعنی `OPERATOR` رنگی کند.

 `HmOperatorPattern.php`

```
1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmOperatorPattern implements Pattern
12 {
13     use IsPattern;
14
15     public function getPattern(): string
16     {
17         return ""; // TODO: Implement
18     }
19
20     public function getTokenType(): TokenTypeEnum
21     {
22         return TokenTypeEnum::TYPE; // TODO: Implement
23     }
24 }
```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به استفاده از عملگرها را به درستی هایلایت کند.

رشته‌ها در زبان برنامه‌نویسی آقای هاشمی به صورت DoubleQuote (" ") مورد استفاده قرار می‌گیرند. شما باید هایلایتر این مورد را به صورتی پیاده‌سازی کنید که محتوای درون این دو علامت نقل‌قول به صورت درستی هایلایت شوند.

```
1 | nam = "Mr ";
2 | famil = "Hashemi";
3 | tedadeBache = 3;
4 | moteahel = 1==1;
5 | mabda = "Kazeroon";
6 | maghsad = "Neishaboor";
7 |
8 | bebin azinja() {
9 |     bechap("Dorood Jahan");
10| }
```

- مثالی از رشته‌ها در زبان هاشمی

شما باید کلاس HmDoubleQuoteValuePattern را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند رشته‌های مختلف را با استفاده از TokenType مربوط به آن یعنی VALUE رنگی کند.

 HmDoubleQuoteValuePattern.php

```
1 | <?php
2 |
3 | declare(strict_types=1);
4 |
5 | namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6 |
7 | use Tempest\Highlight\IsPattern;
8 | use Tempest\Highlight\Pattern;
9 | use Tempest\Highlight\Tokens\TokenTypeEnum;
10|
11| final readonly class HmDoubleQuoteValuePattern implements Pattern
12| {
13|     use IsPattern;
14|
15| }
```

```

15 public function getPattern(): string
16 {
17     return ""; // TODO: Implement
18 }
19
20 public function getTokenType(): TokenTypeEnum
21 {
22     return TokenTypeEnum::TYPE; // TODO: Implement
23 }
24 }

```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به استفاده از رشته‌ها را به درستی هایلایت کند.

▼ ساختار رشته‌های چندخطی در زبان هاشمی و پیاده‌سازی
 کلاس HmTripleDoubleQuoteStringPattern

رشته‌های چندخطی در زبان برنامه‌نویسی آقای هاشمی به صورت TripleDoubleQuote ("" "") مورد استفاده قرار می‌گیرند. شما باید هایلایتر این مورد را به صورتی پیاده‌سازی کنید که محتوای درون این سه‌تایی‌های علامت نقل‌قول به صورت درستی هایلایت شوند.


```

1 bebin azinja() {
2     matn_toolani = "" شعر از مولانا ""
3     خنک آن قمار بازی که بباخت آنچه بودش
4     بنماند هیچش الا هوس قمار دیگر
5     """;
6     bechap(matn_toolani);
7 }

```

- مثالی از رشته‌ها در زبان هاشمی

شما باید کلاس HmTripleDoubleQuoteStringPattern را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند رشته‌های بلند مختلف را با استفاده از TokenType مربوط به آن یعنی VALUE رنگی کند.

 HmTripleDoubleQuoteStringPattern.php

```
1  <?php
2
3  declare(strict_types=1);
4
5  namespace Tempest\Highlight\Languages\Hashemi\Patterns;
6
7  use Tempest\Highlight\IsPattern;
8  use Tempest\Highlight\Pattern;
9  use Tempest\Highlight\Tokens\TokenTypeEnum;
10
11 final readonly class HmTripleDoubleQuoteStringPattern implements Pattern
12 {
13     use IsPattern;
14
15     public function getPattern(): string
16     {
17         return ""; // TODO: Implement
18     }
19
20     public function getTokenType(): TokenTypeEnum
21     {
22         return TokenTypeEnum::TYPE; // TODO: Implement
23     }
24 }
```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به استفاده از رشته‌های بلند را به درستی هایلایت کند.

▼ ساختار تزریق Json در زبان هاشمی و پیاده‌سازی کلاس JsonInjection

همانطور که پیش‌تر در بخش بررسی بیشتر کد هایلایتر به تزریق‌ها اشاره کردیم، در زبان برنامه‌نویسی آقایی هاشمی می‌توان از Json در بین کدهای هاشمی به شکل زیر و با استفاده از farzand z پیاده‌سازی کرد:

```

1 | bebin azinja() {
2 |     object = jfarzand("{"name":"Mr.Hashemi"}");
3 |     bechap(object.name);
4 |     bechap(object.x);
5 | }

```

• مثالی از تزریق Json در زبان هاشمی

شما باید کلاس JsonInjection را به صورت زیر در فایلی با همین نام پیاده‌سازی کنید تا کد هایلایتر بتواند تزریق‌های مختلف Json را در زبان هاشمی را هایلایت کند. در ساختار تزریق‌ها نیازی به TokenType نیست و شما باید در ساختار زیر صرفاً از هایلایتر از پیش ساخته شده‌ی Json درون همین پکیج استفاده کنید.

 JsonInjection.php

```
<?php
```

```
declare(strict_types=1);
```

```
namespace Tempest\Highlight\Languages\Hashemi\Injections;
```

```
use Tempest\Highlight\Highlighter;
```

```
use Tempest\Highlight\Injection;
```

```
use Tempest\Highlight\IsInjection;
```

```
final readonly class JsonInjection implements Injection
```

```
{
```

```
    use IsInjection;
```

```
    public function getPattern(): string
```

```
    {
```

```
        return ""; // TODO: Implement
```

```
    }
```

```
    public function parseContent(string $content, Highlighter $highlighter)
```

```
    {
```

```
23 |         return $highlighter->parse($content, 'json');  
24 |     }  
    }
```

در نهایت در صورتی که کد شما به درستی کار کند، باید مطابق تصویر زیر بخش مربوط به استفاده از تزریق Json را به درستی هایلایت کند.

آنچه باید آپلود کنید

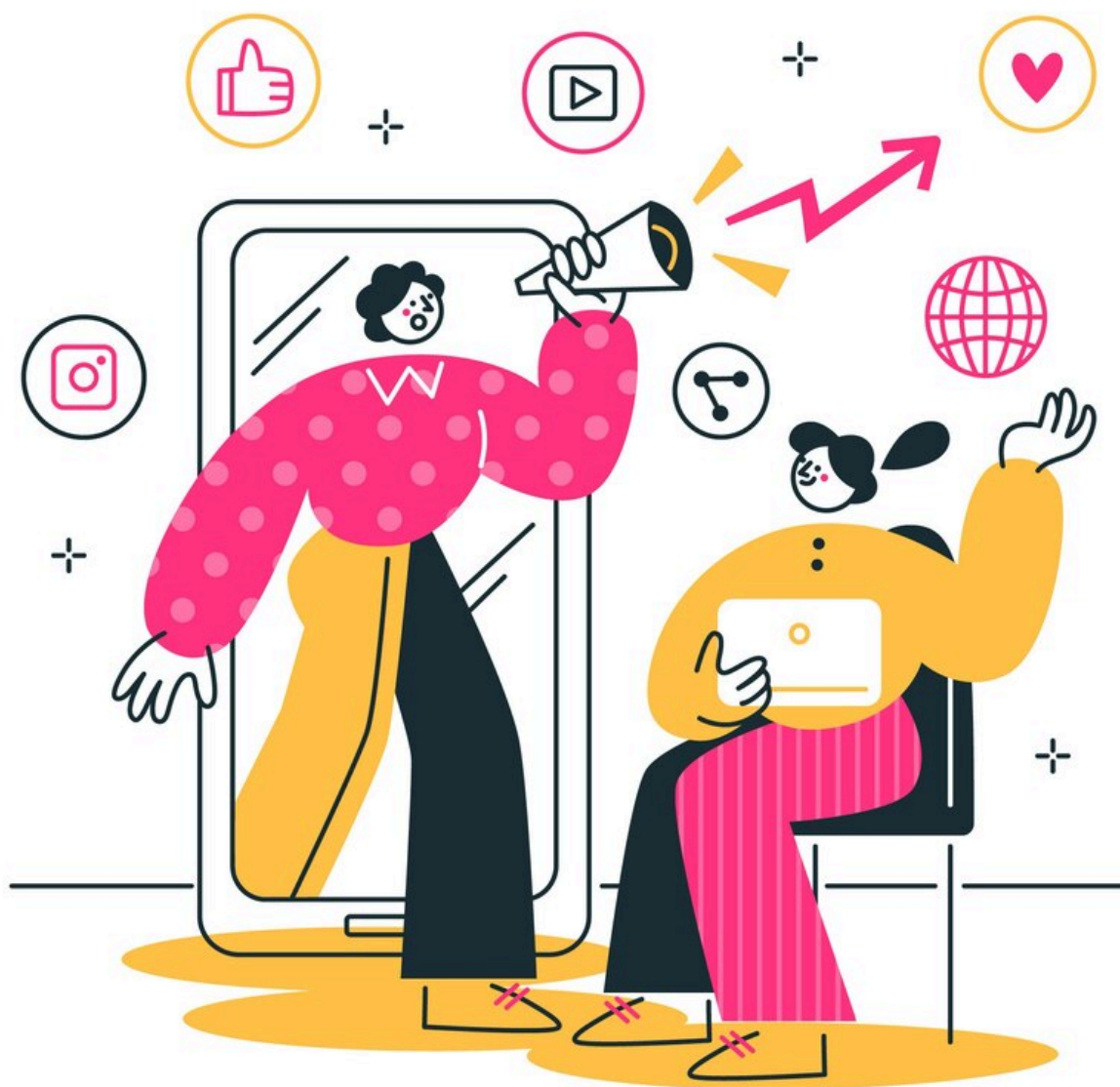
- **توجه:** مطالعه بخش "برسی بیشتر یک مثال مهم در پیاده سازی الگوها (Patterns)" از اهمیت بالایی از جهت نحوه پیاده سازی الگوهای مختلف دارد، لذا در خواندن آن کوشا باشید.
- **توجه:** داوری این سوال در تست های مختلف کدهایی شامل ساختارهای مختلف را به صورت ترکیبی به هایلایتر شما می دهد. دریافت نمره بیشتر از سوال با نسبتی از سوال که ساختار آن ها را کاملاً پیاده سازی کرده اید رابطه مستقیم دارد.
- **توجه:** تنها فایل هایی از پروژه هایلایتر شما در سیستم داوری مورد پذیرش قرار خواهد گرفت که در بخش "ساختار فایل ها" به صورت رنگی مشخص شده است. سایر تغییرات در سایر فایل ها بی تاثیر خواهند بود.
- **توجه:** پس از اعمال تغییرات، کل پروژه به غیر از پوشه ی vendor را ZIP کرده و آپلود کنید.

توجه: در صورتی که فایل ZIP شما بیش از 100MB باشد، سیستم به خودی خود آن را حذف می کند.

پیامرسان پردیس گرام

دوین پس از طراحی موفقیت‌آمیز تمامی سوالات مسیر PHP/Laravel مسابقات /المپیک فناوری، به دلیل مهارت زیاد مسئولیت جدیدی از سمت کوئرا و پارک فناوری پردیس دریافت کرده است، طراحی پیامرسان داخلی پردیس گرام برای برقراری ارتباط راحت‌تر و سریع‌تر مخصوص شرکت‌کنندگان در این سری مسابقات!

شما در این سوال قرار است سراغ پیاده‌سازی بخش کوچکی از این پیامرسان با استفاده از لاراول بپردازید. از آنجایی که رابطه دوین با **رابطه‌های بین مدل‌ها** حسابی شکرآب است، مسئولیت پیاده‌سازی این رابطه‌ها به صورتی که دوین از قبل مشخص کرده است بر عهده شما خواهد بود.



جزئیات پروژه

پروژه‌ی اولیه را از این لینک دانلود کنید.

ساختار فایل‌ها ▼

pardis-messenger

├─ app

| └─ Console

```
|   ├── Exceptions
|   ├── Http
|   ├── Models
|   |   ├── Message.php
|   |   ├── Post.php
|   |   ├── Comment.php
|   |   ├── Like.php
|   |   ├── Profile.php
|   |   ├── Group.php
|   |   ├── Channel.php
|   |   └── User.php
|   └── Providers
├── bootstrap
├── config
├── database
|   ├── factories
|   ├── migrations
|   |   ├── 2014_10_12_000000_create_users_table.php
|   |   ├── 2014_10_12_100000_create_password_resets_table.php
|   |   ├── 2019_08_19_000000_create_failed_jobs_table.php
|   |   ├── 2021_03_18_133215_create_messages_table.php
|   |   ├── 2021_03_18_133224_create_posts_table.php
|   |   ├── 2021_03_18_133232_create_comments_table.php
|   |   ├── 2021_03_18_133241_create_likes_table.php
|   |   ├── 2021_03_18_133250_create_groups_table.php
|   |   └── 2021_03_18_133255_create_channels_table.php
|   └── seeders
├── public
├── resources
├── routes
├── storage
├── tests
├── README.md
├── artisan
├── composer.json
├── composer.lock
├── package.json
├── phpunit.xml
```

└─ server.php
└─ webpack.mix.js

▼ راه‌اندازی پروژه

برای اجرای پروژه، باید php و composer را از قبل نصب کرده باشید.

- ابتدا پروژه‌ی اولیه را دانلود و از حالت فشرده خارج کنید.
- دستور composer install را در پوشه‌ی اصلی پروژه برای نصب نیازمندی‌ها اجرا کنید.
- برای اجرای مایگريشن‌ها از دستور php artisan migrate استفاده کنید.

▼ ساختار جداول

مایگريشن‌های مربوط به پروژه از قبل ایجاد شده‌اند. در اولین گام باید ساختار جداول پروژه را مطابق توضیحات زیر تکمیل کنید:

ساختار جدول users :

نام ستون	نوع	تعریف
id	bigint	کلید اصلی
name	string	نام کاربر
email	string	ایمیل کاربر (منحصربه‌فرد)
email_verified_at	timestamp	زمان تایید ایمیل
remember_token	string	حفظ اطلاعات ورود کاربر
password	string	رمز عبور کاربر
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول : profiles

نام ستون	نوع	تعریف
id	bigint	کلید اصلی
user_id	bigint	کلید خارجی به جدول users
bio	text	بیوگرافی کاربر
avatar	string	آدرس تصویر پروفایل
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول : messages

نام ستون	نوع	تعریف
id	bigint	کلید اصلی
user_id	bigint	کلید خارجی به جدول users
content	text	محتوای پیام
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول : posts

نام ستون	نوع	تعریف
id	bigint	کلید اصلی

نام ستون	نوع	تعريف
user_id	bigint	کلید خارجی به جدول users
channel_id	bigint	کلید خارجی به جدول channels
content	text	محتوای پست
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول comments :

نام ستون	نوع	تعريف
id	bigint	کلید اصلی
content	text	محتوای کامنت
user_id	bigint	کلید خارجی به جدول users
commentable_id	bigint	شناسه موجودیت (پست یا پیام)
commentable_type	string	نوع موجودیت (message یا post)
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول likes :

نام ستون	نوع	تعريف
id	bigint	کلید اصلی

نام ستون	نوع	تعريف
user_id	bigint	کلید خارجی به جدول users
likeable_id	bigint	شناسه موجودیت (پست یا کامنت)
likeable_type	string	نوع موجودیت (comment یا post)
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول groups :

نام ستون	نوع	تعريف
id	bigint	کلید اصلی
name	string	نام گروه
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول group_user :

نام ستون	نوع	تعريف
group_id	bigint	کلید خارجی به جدول groups
user_id	bigint	کلید خارجی به جدول users

ساختار جدول channels :

نام ستون	نوع	تعریف
id	bigint	کلید اصلی
name	string	نام چنل
created_at	timestamp	زمان ایجاد
updated_at	timestamp	زمان آخرین به‌روزرسانی

ساختار جدول channel_user :

نام ستون	نوع	تعریف
channel_id	bigint	کلید خارجی به جدول channels
user_id	bigint	کلید خارجی به جدول users

شما در این سوال صرفاً قرار است تا روابط بین مدل‌های پیام‌رسان را پیاده‌سازی کنید. مایگريشن‌ها از قبل پیاده‌سازی شده‌اند و در قالب پروژه اولیه به همراه توضیحات آن‌ها در بخش ساختار جداول در اختیار شما قرار داده شده‌اند. توضیحات روابطی که باید پیاده‌سازی شوند به شکل زیر است:

مدل‌های پیام‌رسان پردیس‌گرام:

لیست مدل‌های این پیام‌رسان به شرح زیر است:

- مدل User : کاربران سیستم
- مدل Profile : پروفایل کاربر
- مدل Message : پیام‌های کاربران
- مدل Post : پست‌های ارسال‌شده توسط کاربران در چنل‌ها
- مدل Comment : کامنت‌های پست‌ها و پیام‌ها
- مدل Like : لایک‌های داده‌شده به پست‌ها و کامنت‌ها

- مدل Group : گروه‌های کاربری شامل کاربران مختلف
- مدل Channel : چنل‌ها که پست‌ها در آن‌ها ارسال می‌شوند

روابط بین مدل‌ها:

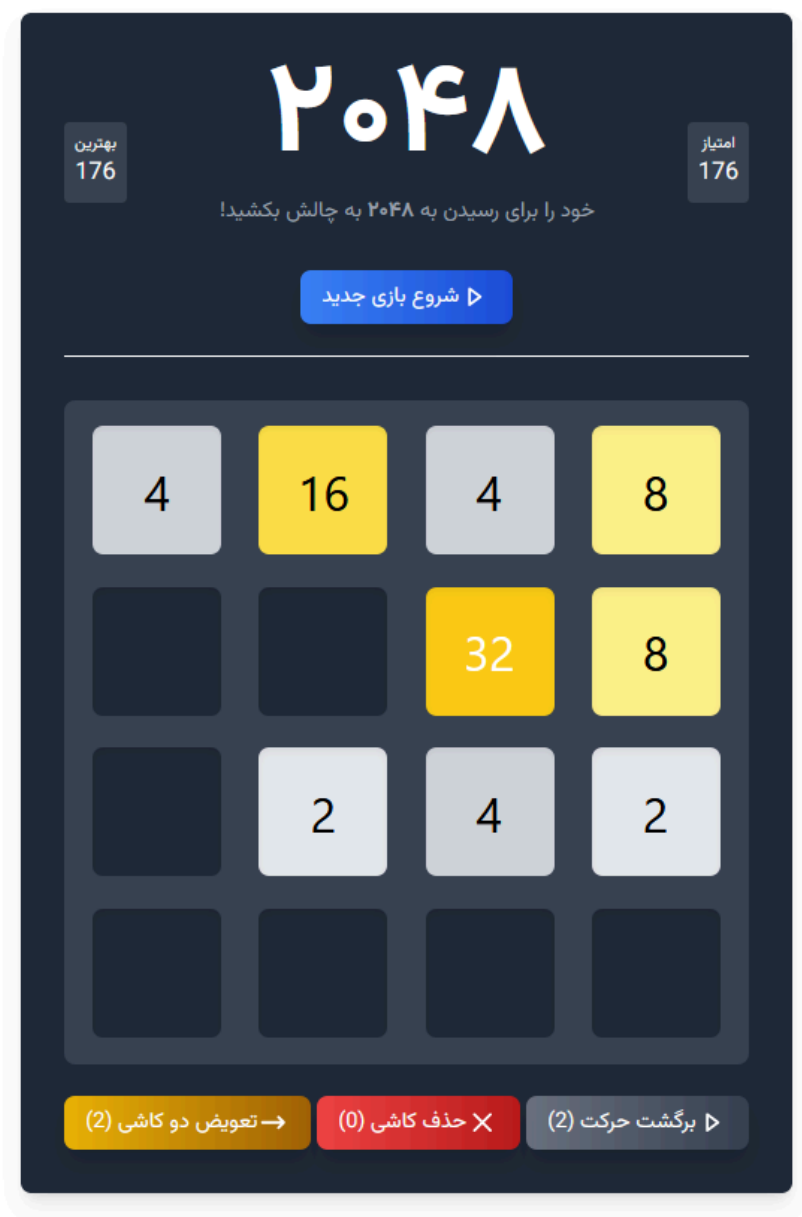
- هر کاربر یک پروفایل دارد.
- هر کاربر می‌تواند چند پیام ارسال کند.
- هر کاربر می‌تواند چند پست ایجاد کند و در چندین چنل عضو باشد.
- هر پست می‌تواند چند کامنت داشته باشد.
- هر پست و کامنت می‌توانند لایک داشته باشد (استفاده از رابطه‌ی پولیمورفیک برای لایک‌ها).
- هر کامنت متعلق به یک پست یا پیام است (رابطه‌ی پولیمورفیک برای کامنت‌ها).
- هر کاربر می‌تواند در چند گروه کاربری عضو باشد (رابطه‌ی چند به چند برای گروه‌ها).
- هر چنل شامل چندین کاربر است و هر کاربر می‌تواند در چند چنل عضو باشد.
- هر پست متعلق به یک چنل است و هر کاربر می‌تواند در چندین چنل پست ارسال کند.

آنچه باید آپلود کنید

- توجه: نیازی به پیاده‌سازی مایگريشن‌ها در این سوال نیست و این مایگريشن‌ها از قبل در اختیار شما قرار داده شده است. وظیفه شما در این سوال تنها پیاده‌سازی روابط مدل‌ها می‌باشد.
- توجه: تنها تغییرات مورد نیاز را در مدل‌ها اعمال کنید تا روابط به‌درستی تعریف شوند.
- توجه: پس از تعریف روابط، کل فایل‌های پروژه به‌جز پوشه‌ی vendor را زیپ کرده و ارسال کنید.
- توجه: که شما مجاز به افزودن فایل جدیدی در این ساختار نیستید و تنها باید تغییرات را در فایل‌های موجود اعمال کنید.
- توجه: که نام فایل ZIP اهمیتی ندارد.

۲۰۴۸ شگفت‌انگیز

علی پس از جابه‌جا کردن مرزهای طراحی مسابقات برنامه‌نویسی در کوئرا، این‌بار تصمیم گرفته تا رشته جدیدی را به #المپیک‌فناوری پردیس اضافه کند که از یک بازی قدیمی که در ایام کودکی بسیار به آن علاقه داشته به شیوه جدیدی در آن استفاده می‌کند. این بازی مورد علاقه علی، بازی معروف ۲۰۴۸ می‌باشد که در دورانی جزو محبوب‌ترین بازی‌های فکری بود. شما در این سوال به پیاده‌سازی یک نسخه شگفت‌انگیز از این بازی خواهید پرداخت که قابلیت‌های عجیبی دارد که در نسخه کلاسیک این بازی یافت نمی‌شود.



جزئیات پروژه

پروژه‌ی اولیه را از این لینک دانلود کنید.

▼ ساختار فایل‌ها

```
1 | moeine-dooz-o-kalak
2 | └─ app
3 | └─ Http
```

```
4 | | | └─ Livewire
5 | | | └─ Game2048.php
6 | | └─ Models
7 | └─ Providers
8 └─ bootstrap
9 └─ config
10 └─ database
11 └─ public
12 └─ resources
13 | └─ views
14 | | └─ livewire
15 | | | └─ game2048.blade.php
16 | | └─ 2048.blade.php
17 | └─ css
18 └─ routes
19 | └─ web.php
20 └─ storage
21 └─ tests
22 └─ README.md
23 └─ artisan
24 └─ composer.json
25 └─ composer.lock
26 └─ package.json
27 └─ phpunit.xml
28 └─ vite.config.js
```

▼ راه اندازی پروژه

برای اجرای پروژه، باید php و composer را از قبل نصب کرده باشید.

- ابتدا پروژهی اولیه را دانلود و از حالت فشرده خارج کنید.
- دستور `composer install` را در پوشه‌ی اصلی پروژه برای نصب نیازمندی‌ها اجرا کنید.
- دستور `npm install` را در پوشه‌ی اصلی پروژه برای نصب نیازمندی‌ها اجرا کنید. (توجه کنید که این پروژه از *Tailwindcss* استفاده می‌کند)
- دستور `npm run dev` را در مسیر پوشه اصلی پروژه اجرا کنید. در صورتی که این دستور را اجرا نکنید نمی‌توانید ویوهای ساخته شده با *Tailwindcss* را مشاهده کنید.

- برای اجرای تست‌های نمونه، می‌توانید از دستور `php artisan test` استفاده کنید.

▼ لینک‌های مفید

در بخش زیر لیستی از لینک‌های مفید وبسایت‌ها و مستندات که می‌تواند به شما در پیاده‌سازی این سوال کمک کند برای شما قرار گرفته است:

- لینک مستندات *Livewire*
- لینک نسخه آنلاین بازی ۲۰۴۸
- لینک ویکی‌پدیا در مورد توضیحات بازی ۲۰۴۸

پیاده‌سازی پروژه

در این سوال قرار است شما به سراغ پیاده‌سازی این بازی شگفت‌انگیز با استفاده از *Livewire3* بروید! بخش‌هایی از این بازی از پیش برای شما پیاده‌سازی شده است و شما صرفاً قرار است تا به سراغ پیاده‌سازی برخی موارد گفته شده در سوال بروید. توجه کنید که این سوال جزئیات فراوانی دارد پس در خواندن متن سوال دقت لازم را داشته باشید.

▼ جزئیات دقیق‌تر و معرفی بازی ۲۰۴۸

جزئیات دقیق‌تر و معرفی بازی ۲۰۴۸

≡ 2048

SCORE
48

BEST
2816

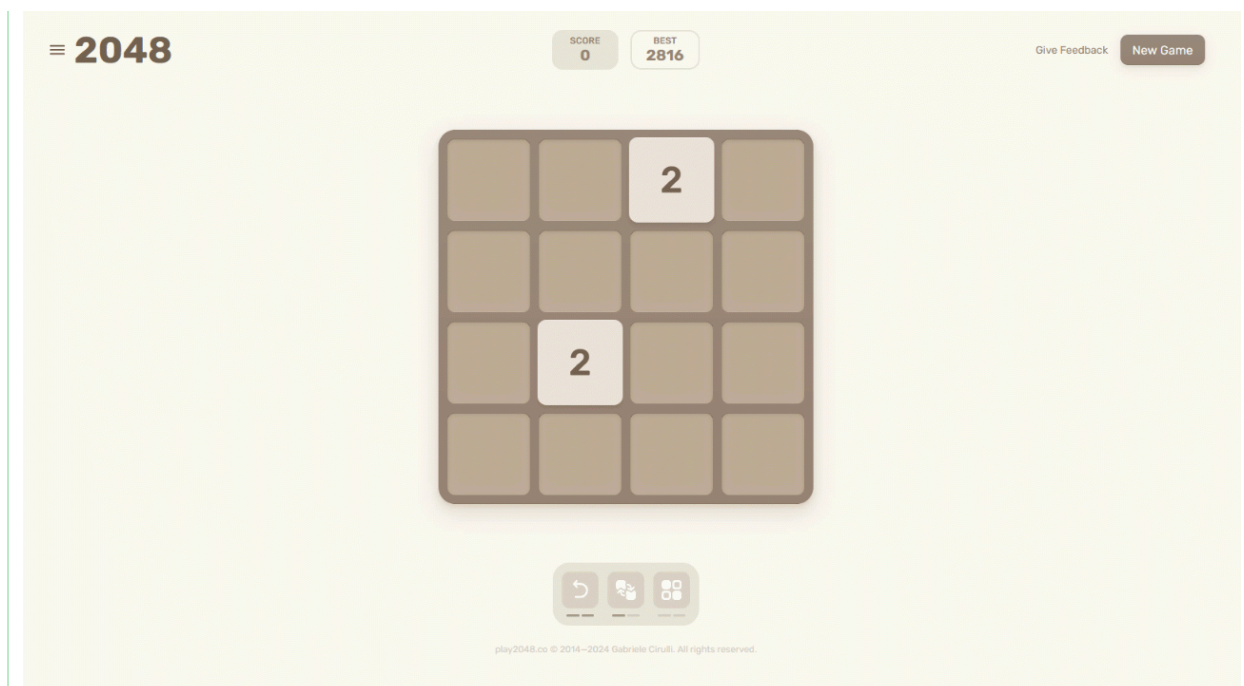
Give Feedback

New Game

4	2	8	8
2			4
			8
			2



بازی معروف ۲۰۴۸ روی یک جدول شطرنجی ۴×۴ ساده با کاشی‌های **شماره‌دار** بازی می‌شود که وقتی بازیکن آن‌ها را با استفاده از **چهار کلید جهت‌دار** حرکت می‌دهد، می‌لغزند. در هر نوبت، یک کاشی به‌طور تصادفی در نقطه‌ای خالی روی تابلو ظاهر می‌شود که مقدار آن ۲ یا ۴ است. کاشی‌ها تا جایی که ممکن است در جهت انتخاب شده می‌لغزند؛ تا زمانی که کاشی دیگر یا لبه شبکه شطرنجی متوقفشان نکند. اگر در حین حرکت، دو کاشی با **شماره یکسان** به هم بخورند، آن‌گاه در یک کاشی که ارزشش برابر با مجموع ارزش دو کاشی برخوردی است، **ادغام** می‌شود. کاشی به‌دست‌آمده **نمی‌تواند** در همان حرکت، دوباره با کاشی دیگری ادغام شود (به عنوان مثال اگر در یک جهت سه عدد کاشی با شماره ۴ یافت می‌شد، پس از انجام یک حرکت دو عدد از آنها با یکدیگر ادغام می‌شود و مقدار ۸ را تشکیل می‌دهند اما کاشی سوم که مقدار ۴ دارد ادغام نشده باقی می‌ماند). کاشی‌های با امتیاز بالاتر، درخششی روشن‌تر از خود ساطع می‌کنند.




- در بخش بالا می‌توانید دمویی از این بازی را مشاهده کنید. همچنین شما می‌توانید این بازی را از طریق [این لینک](#) به صورت آنلاین بر روی مرورگر خود بازی کنید.

▼ جزئیات و ساختار Route و View بازی

برای دسترسی به این بازی در پروژه اولیه یک Route با آدرس `/2048` به صورت پیشفرض تعریف شده است که به ویو `2048` متصل شده است. در این مسیردهی و ویو تعریف شده نباید تغییری داده شود.

در فایل مربوط به ویو `2048` شما باید **کامپوننتی (Component)** که از قبل در ساختار پروژه اولیه با نام `game2048` تعریف شده است را **استفاده (Include)** کنید. این کامپوننت همان بخش اصلی بازی دوز است که قرار است تا اجزای مختلف کامپوننتش را در بخش‌های بعدی این سوال پیاده‌سازی کنید تا در نهایت بازی به درستی اجرا شود.

 laravel

```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5
```

```

5 Route::get('/2048', function () {
6     return view('2048');
7 });

```

 laravel

```

1 <!DOCTYPE html>
2 <html lang="fa" dir="rtl">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>🎮2048 Shegeft Angiz🎮</title>
7     <link href="https://cdn.jsdelivr.net/gh/rastikerdar/vazirmatn@v33"
8         @vite('resources/css/app.css')
9         @livewireStyles
10 </head>
11 <body>
12
13 {{--TODO: Include Laravel game component--}}
14
15 @livewireScripts
16 </body>
17 </html>

```

و در نهایت ساختار View مربوط به کامپوننت بازی که در مسیر
resources/views/livewire/game2048.blade.php قرار گرفته است و باید در ویو
2048.blade.php از آن استفاده شود به شکل زیر است:

 laravel

```

<div class="max-w-xl mx-auto text-center text-gray-100 bg-gray-800 p-4">
    <div class="flex justify-between items-center mb-6">
        <!-- Player score section -->
        <div class="bg-gray-700 text-white p-2 rounded mt-6">
            <span class="block text-xs font-vazirmatn">امتیاز</span>

```

```

        <span class="text-lg font-vazirmatn">0</span>
    </div>
    <!-- Player score section -->

    <div class="text-white">
        <h1 class="text-8xl font-bold mb-1 font-vazirmatn">۲۰۴۸</h1>
        <p class="my-2 mx-2 text-gray-400 font-vazirmatn">سیدن به</p>
    </div>

    <!-- Highest score section -->
    <div class="bg-gray-700 text-white p-2 rounded mt-6">
        <span class="block text-xs font-vazirmatn">بهترین</span>
        <span class="text-lg font-vazirmatn">0</span>
    </div>
    <!-- Highest score section -->
</div>

<div class="flex justify-center items-center mb-6">
    <!-- Reset game button -->
    <button class="font-vazirmatn bg-gradient-to-r from-blue-500 to-blue-600 text-white p-2 rounded" type="button">
        <svg class="w-6 h-6 text-blue-800 dark:text-white" aria-hidden="true">
            <path stroke="currentColor" stroke-linecap="round" stroke-width="2">
                </path>
        </svg>
        شروع بازی جدید
    </button>
    <!-- Reset game button -->
</div>

<div class="my-4">
    <hr class="border-t border-white" />
</div>

<!-- Game message Section -->
<div class="mt-4 text-center text-lg text-green-600 font-vazirmatn">
    <!-- Show game message -->
</div>
<!-- Game message Section -->

<!-- Game grid Section -->
<div class="grid grid-cols-4 gap-4 bg-gray-700 p-4 rounded-lg relative">

```

```

<!-- Grid cells Section -->
<div class="text-4xl h-24 w-24 flex items-center justify-center"
  @if ($cell === 0) bg-gray-800 text-transparent
  @elseif ($cell === 2) bg-gray-200 text-black
  @elseif ($cell === 4) bg-gray-300 text-black
  @elseif ($cell === 8) bg-yellow-200 text-black
  @elseif ($cell === 16) bg-yellow-300 text-black
  @elseif ($cell === 32) bg-yellow-400 text-white
  @elseif ($cell === 64) bg-orange-300 text-white
  @elseif ($cell === 128) bg-orange-400 text-white
  @elseif ($cell === 256) bg-orange-500 text-white
  @elseif ($cell === 512) bg-red-300 text-white
  @elseif ($cell === 1024) bg-red-400 text-white
  @elseif ($cell === 2048) bg-red-500 text-white
  @endif">
</div>
<!-- Grid cells Section -->

<!-- Gameover and restart Section -->
<div class="absolute inset-0 flex flex-col items-center justify-center"
  <h2 class="text-3xl font-bold text-white mb-4 font-vazirmatn">
  <button class="mt-4 px-4 py-2 bg-white text-gray-800 rounded">
</div>
<!-- Gameover and restart Section -->
</div>
<!-- Game grid Section -->

<div class="flex justify-between items-center mt-6">
  <!-- Undo action button -->
  <button class="font-vazirmatn bg-gradient-to-r from-gray-500 to-gray-600"
    <svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true">
      <path stroke="currentColor" stroke-linecap="round" stroke-width="2">
    </svg>
  </button>
  <!-- Undo action button -->

  <!-- Delete tile button -->

```

برگشت حرکت (0)

```

        <button class="font-vazirmatn bg-gradient-to-r from-red-500 to-red-800" type="button">
            <svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true">
                <path stroke="currentColor" stroke-linecap="round" stroke-width="2">
                    </path>
            </svg>
        </button>
    <!-- Delete tile button -->

    <!-- Swap tile button -->
    <button class="font-vazirmatn bg-gradient-to-r from-yellow-500 to-yellow-800" type="button">
        <svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true">
            <path stroke="currentColor" stroke-linecap="round" stroke-width="2">
                </path>
        </svg>
    </button>
    <!-- Swap tile button -->
</div>

<script>
    document.addEventListener('keydown', function (event) {
        let direction = '';

        switch (event.key) {
            case 'ArrowUp':
                direction = 'up';
                @this.move('up');
                break;
            case 'ArrowDown':
                direction = 'down';
                @this.move('down');
                break;
            case 'ArrowLeft':
                direction = 'left';
                @this.move('left');
                break;
            case 'ArrowRight':
                direction = 'right';
                @this.move('right');
                break;
        }
    });

```

```
});  
</script>  
</div>
```

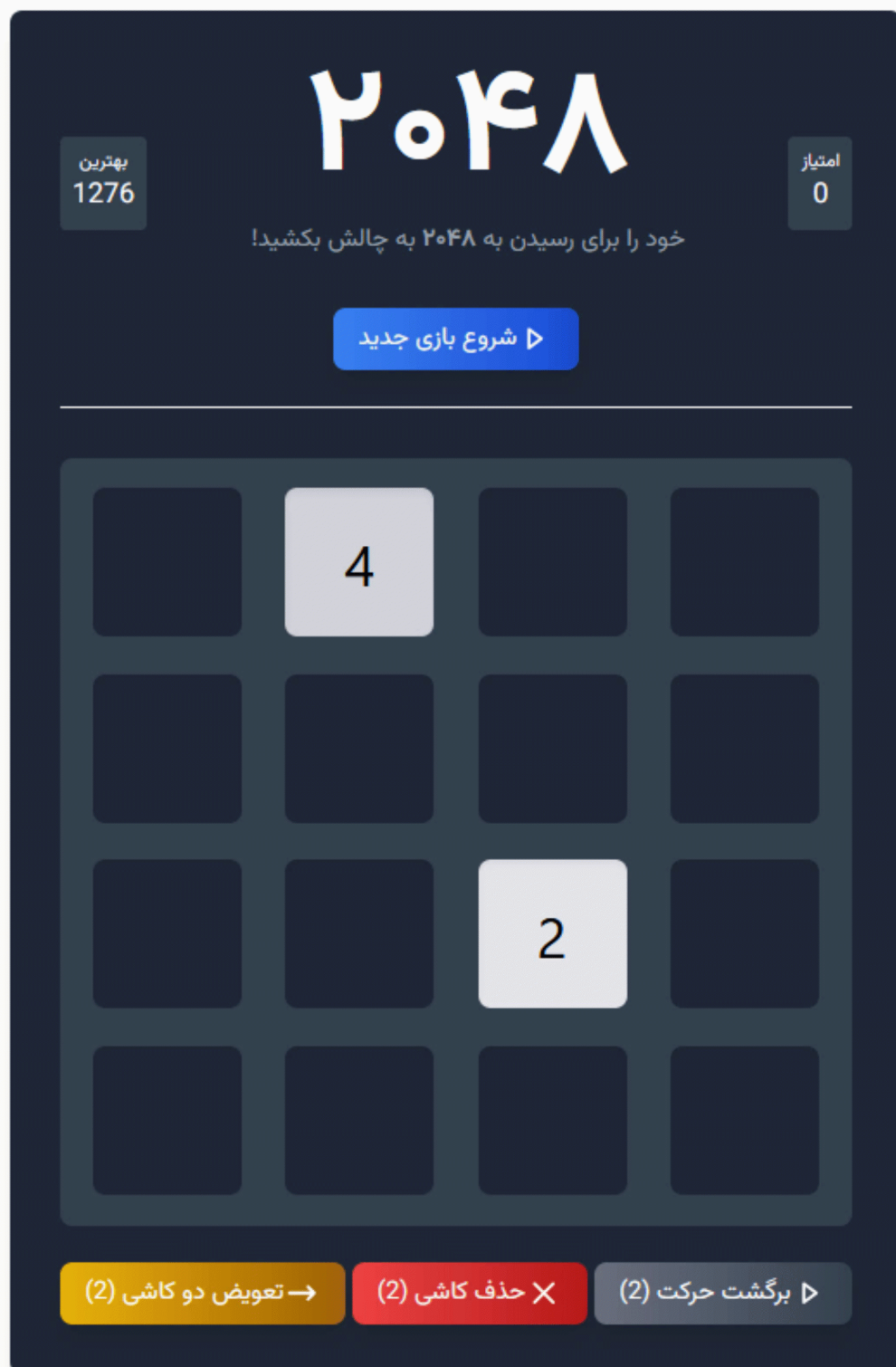
- این ساختار صرفاً ساختار آماده مربوط به رابط گرافیکی بازی می‌باشد و منطق نمایش المان‌های مختلف آن باید توسط شما در این کامپوننت پیاده‌سازی شود.

▼ ساختار کلی بازی و وضعیت‌های برد و باخت

ساختار کلی بازی

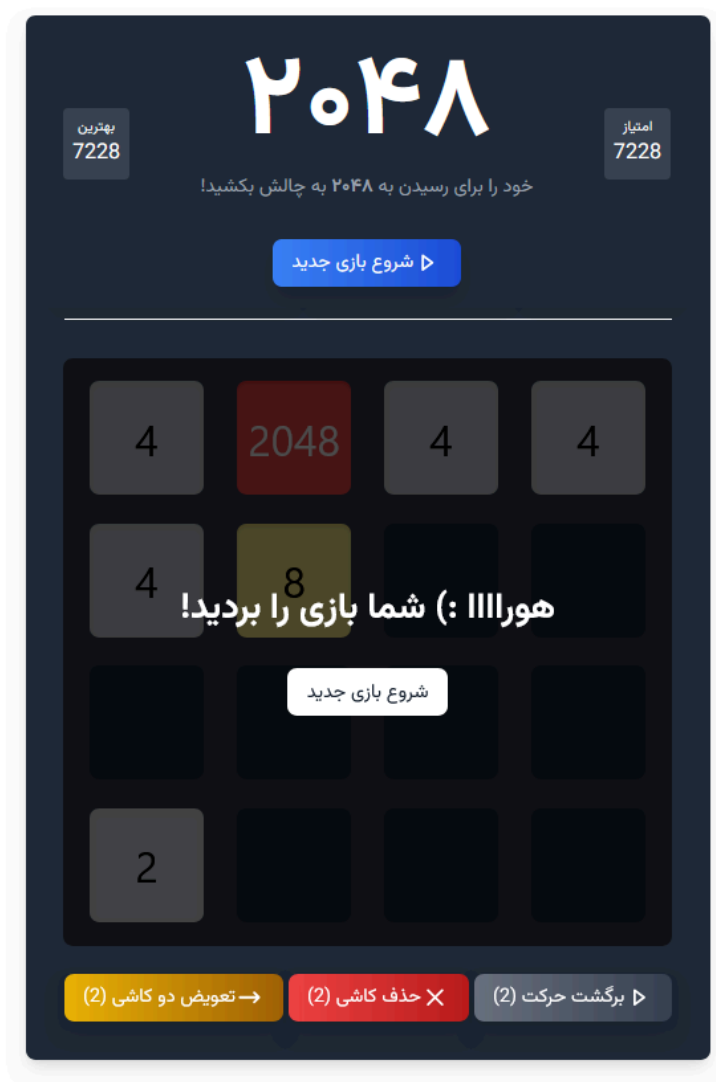
ساختار کلی این بازی بسیار شبیه به **نسخه اصلی بازی ۲۰۴۸** می‌باشد. در هر وضعیت آغازین ۲ عدد کاشی به صورت تصادفی در دو تا از خانه‌های جدول قرار می‌گیرند. این مقادیر تصادفی باید حتما مقدار ۲ و یا ۴ باشد. پس از وضعیت آغازین فقط در هر مرتبه بازی در صورتی که جدول دچار تغییراتی شود، یک کاشی جدید دیگر که دارای مقدار ۲ و یا ۴ است به صورت تصادفی در جایی از جدول اضافه خواهد شد. در صورتی که هیچ تغییری در جدول ایجاد نشود، هیچ کاشی جدیدی در بازی اضافه نمی‌شود.

همچنین توجه داشته باشید که در صورت استفاده از قابلیت‌های شگفت‌انگیز بازی نظیر برگشت به حرکت قبل و یا حذف کاشی و تعویض دو کاشی **هیچ کاشی جدیدی** به جدول اضافه نخواهد شد.



وضعیت برد

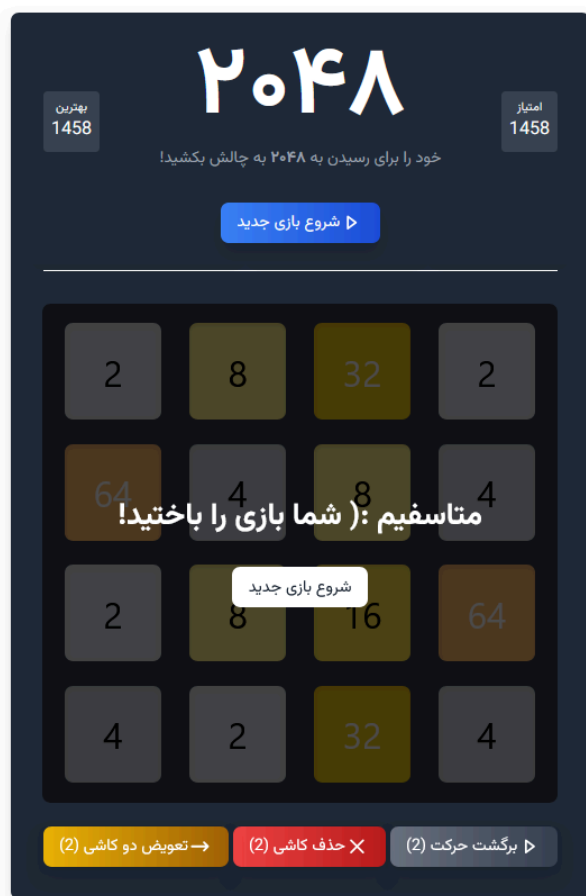
یک وضعیت برد زمانی برای شما اتفاق می‌افتد که شما بتوانید با تعدادی حرکت حداقل یک کاشی با ارزش ۲۰۴۸ ایجاد کنید. در این صورت بازی به پایان می‌رسد و علاوه بر نمایش پیام "هورااااا! (: شما بازی را بردید!" باید از کاربر دعوت شود تا دوباره بازی جدیدی را آغاز کند. (توجه کنید که متن مشخص شده برای وضعیت برد نباید تغییری کند و باید دقیقا عبارت گفته شده باشد.)



وضعیت باخت

یک وضعیت باخت زمانی برای شما اتفاق می‌افتد که شما پس از انجام تعدادی حرکت تمام صفحه را با کاشی‌هایی مقادیر مختلف پر کردید که دیگر نمی‌توانید هیچ عمل ادغام دو کاشی‌ای را انجام دهید. در این حالت شما هر حرکتی انجام دهید هیچ تغییری در صفحه ایجاد نمی‌شود، همچنین شما قادر به ساخت هیچ

کاشی‌ای با ارزش ۲۰۴۸ نیز نشده‌اید. در این حالت شما بازی را خواهید باخت و به شکل زیر پیام "متاسفیم:" شما بازی را باختید!" برای شما نشان داده خواهد شد و از شما دعوت به عمل می‌آید تا بازی جدیدی را آغاز کنید. (توجه کنید که متن مشخص شده برای وضعیت باخت نباید تغییری کند و باید دقیقا عبارت گفته شده باشد.)



موردی که قابل توجه این است که صفحه‌ی بازی، امتیاز و تمامی مقادیر دیگر مثل بیشترین امتیاز یا تعداد نوبت باقی مانده از قابلیت‌های شگفت‌انگیز باید در تمامی مراحل در قالب نشست (Session) ذخیره شود تا در صورتی که صفحه بسته شد، بازی قابلیت ادامه دادن در آینده را داشته باشد.

همچنین توجه کنید که امتیاز فعلی و بیشترین امتیاز همواره به صورت زنده در صفحه بازی بروزرسانی می‌شود. امتیاز هر بازی در ابتدا صفر است و در صورتی که دو کاشی با یکدیگر ادغام شوند، امتیاز به اندازه مقدار کاشی جدید که پس از ادغام بدست آمده است اضافه خواهد شد. همچنین در صورتی که امتیاز فعلی

بازی از بیشترین امتیاز بدست آمده بیشتر بود باید مقدارش با آن جایگزین شود تا **همواره بیشترین امتیاز کسب شده** در تمام بازی‌ها در صفحه بازی نمایش داده شود.

▼ ساختار بخش قابلیت‌های شگفت‌انگیز بازی

بخش متفاوت و شگفت‌انگیز این بازی نسبت به حالت کلاسیک بازی ۲۰۴۸ در این بخش خلاصه می‌شود. به تصویر زیر دقت کنید:



بخش قابلیت‌های شگفت‌انگیز بازی از سه بخش "برگشت حرکت"، "حذف کاشی" و "تعویض دو کاشی" تشکیل شده است که عملکردی آن‌ها به شکل زیر است:

- **بخش "برگشت حرکت":** با فشردن این دکمه بازی به یک حرکت قبل بر می‌گردد. در واقع این دکمه همان دکمه آندو (Undo) بازی می‌باشد که در صورتی که کاربر حرکت اشتباهی را انجام دهد می‌تواند جهت جبران آن از این قابلیت استفاده کند.
- **بخش "حذف کاشی":** با فشردن این دکمه کاربر می‌تواند با انتخاب یکی از کاشی‌های صفحه آن را به کل از صفحه حذف نماید. این کار اجازه می‌دهد تا با حذف کاشی یا کاشی‌های مزاحم کاربر بتواند بازی را به نفع خود جلو ببرد.
- **بخش "تعویض دو کاشی":** با فشردن این دکمه، کاربر باید از صفحه بازی دو کاشی را انتخاب کند. دو کاشی انتخاب شده جای خود را با یکدیگر تغییر می‌دهند. از این قابلیت می‌توان در جهت برد بازی در

حالاتی که نیاز است دو کاشی با ارزش یکسان در کنار یکدیگر قرار بگیرند تا ادغام شوند و کاشی با ارزش‌تری را بسازند استفاده کرد.

تمامی این قابلیت‌های شگفت‌انگیز به کاربر اجازه می‌دهد تا فرصت جبران حرکت اشتباه و یا تغییر بازی به وضعیتی را داشته باشد که مطلوب‌تر است و باعث کسب امتیاز بیشتری خواهد شد.

توجه شود که کاربر در کل نشست (*Session*) خود تنها اجازه ۲ نوبت استفاده از هر کدام از این قابلیت‌ها را خواهد داشت و در صورتی که این نوبت‌ها به اتمام برسد، کاربر دیگر اجازه استفاده از این قابلیت‌ها را نخواهد داشت. همچنین هر مرتبه که کاربر بتواند یک کاشی جدید با ارزش ۵۱۲ بسازد، یک نوبت جدید برای استفاده از هر کدام از این قابلیت‌های شگفت‌انگیز به او افزوده خواهد شد. حداکثر مقدار نوبت‌های مجاز برای تمامی این قابلیت‌ها، ۲ نوبت می‌باشد و کاربر می‌تواند این نوبت‌ها را برای باقی بازی‌های خود نیز نگهداری کند. (یعنی این نوبت‌ها در هر نشست تعریف می‌شوند نه در هر بازی و با اتمام بازی تعداد نوبت‌ها ریست نخواهد شد)

۲۰۴۸

بهترین
1584

امتیاز
800

خود را برای رسیدن به ۲۰۴۸ به چالش بکشید!

▶ شروع بازی جدید

4	16	16	8
4	4	128	4
	4	8	8
			4

→ تعویض دو کاشی (1)

✕ حذف کاشی (2)

▶ برگشت حرکت (2)

با توجه به موارد گفته شده در بخش‌های بالا، شما باید ساختار کامپوننت *Livewire* ای *Game2048* را که در ساختار پروژه اولیه در مسیر `app/Livewire/Game2048.php` قرار دارد را مطابق با ساختار زیر پیاده سازی کنید:

 laravel

```
1  <?php
2
3  namespace App\Livewire;
4
5  use Livewire\Component;
6
7  class Game2048 extends Component
8  {
9      public $grid = [];
10     public $score = 0;
11     public $highScore = 0;
12
13     public $undoUses = 2;
14     public $swapUses = 2;
15     public $deleteTileUses = 2;
16
17     public function mount()
18     {
19         // TODO: Implement
20     }
21
22     public function render()
23     {
24         // TODO: Implement
25     }
26
27     public function move($direction)
28     {
29         // TODO: Implement
30     }
31
32     public function undoAction()
33     {
```

```

33     {
34         // TODO: Implement
35     }
36
37     public function deleteTile($x, $y)
38     {
39         // TODO: Implement
40     }
41
42     public function selectTileForSwap($x, $y)
43     {
44         // TODO: Implement
45     }
46
47     public function swapTiles()
48     {
49         // TODO: Implement
50     }
51
52     public function resetGame()
53     {
54         // TODO: Implement
55     }
56 }

```

توضیحات ساختار و ویژگی‌های این کامپوننت به شکل زیر است:

- **متغیر grid** : این متغیر همان‌طور که از اسمش پیداست ساختار جدول را در خود ذخیره می‌کند. هر کدام از خانه‌های این جدول می‌توانند 0 (به معنی خانه‌ی خالی و پوچ بدون کاشی) و یا توانی از دو باشند که مقدار ارزش کاشی آن خانه را مشخص خواهد کرد. توجه کنید که اندیس‌های جدول باید همگی از 0 شروع شوند.
- **متغیر score** : این متغیر نشان‌دهنده امتیاز فعلی کاربر در بازی می‌باشد. امتیاز کاربر و جدول بازی با هر بار ریلود شدن صفحه نباید تغییری کنند.
- **متغیر highScore** : این متغیر مشخص می‌کند که بالاترین امتیاز کسب شده در بازی توسط کاربر در تمامی بازی‌هایی که انجام داده است چه مقداری است.

- **متغیرهای** `undoUses` ، `swapUses` و `deleteTileUses` : این متغیرها همانطور که از نامشان پیداست نشان‌دهنده تعداد هر کدام از ویژگی‌های شگفت‌انگیز است. همانطور که پیش‌تر توضیح داده شد؛ هر کاربر در تمام نشست خود تنها اجازه 2 بار استفاده از هر کدام از این ویژگی‌ها را دارد. همچنین فرصت استفاده از این ویژگی‌ها با توجه به توضیحات گفته شده می‌تواند تحت شرایطی افزایش یابد.

توابعی که شما باید در این سوال پیاده‌سازی کنید به شکل زیر هستند:

- **تابع** `mount` : این تابع در واقع همان تابع سازنده کامپوننت خواهد بود.
- **تابع** `move` : این تابع تقریباً مهم‌ترین بخش بازی می‌باشد. یک آرگومان ورودی دریافت می‌کند که یکی از مقادیر `up` ، `down` ، `left` و `right` است که نشان‌دهنده جهت حرکت است. این مقدار توسط اسکریپت `Js` که در انتهای ویوی بازی قرار دارد و مسئول تشخیص دکمه‌های حرکتی بازی است به این تابع ورودی داده خواهد شد.
- **تابع** `undoAction` : این تابع مسئول برگشت حرکت به حرکت قبلی است و هیچ آرگمانی را به عنوان ورودی دریافت نمی‌کند. در صورتی که تعداد فرصت‌های استفاده از این قابلیت **صفر نباشد**، با فراخوانی این تابع ساختار جدول بازی به یک حرکت قبل از حرکت فعلی باز خواهد گشت.
- **تابع** `deleteTile` : با هر بار اجرا شدن این تابع که دو مقدار ورودی `x` و `y` دارد که به ترتیب موقعیت **سطر** و **ستون** خانه‌ای که قصد داریم کاشی از آن حذف شود را مشخص می‌کنند، در صورتی که آن کاشی وجود داشته باشد (*مقدار 0 نداشته باشد*) و تعداد فرصت‌های استفاده از این قابلیت **صفر نباشد**، کاشی مورد نظر را حذف خواهد کرد.
- **تابع** `selectTileForSwap` : با هر بار اجرا شدن این تابع که دو مقدار ورودی `x` و `y` دارد که به ترتیب موقعیت **سطر** و **ستون** خانه‌ای که قصد داریم کاشی‌اش را انتخاب کنیم را مشخص می‌کنند، در صورتی که آن کاشی وجود داشته باشد (*مقدار 0 نداشته باشد*) آن را انتخاب خواهد کرد. انتخاب شدن به صورت ذخیره‌سازی موقعیت این کاشی خواهد بود تا با اجرای دوباره‌ی آن بتوان کاشی دومی را نیز انتخاب و موقعیت آن را ذخیره‌سازی کرد.
- **تابع** `swapTiles` : این تابع با استفاده از مقادیر ذخیره‌سازی شده با استفاده از `selectTileForSwap` در هنگام فراخوانی در صورتی که تعداد فرصت‌های استفاده از این قابلیت **صفر نباشد**، موقعیت دو کاشی را با هم تعویض می‌کند. این تابع هیچ آرگومان ورودی‌ای ندارد.
- **تابع** `render` : این تابع کامپوننت بازی را رندر می‌کند.

آنچه باید آپلود کنید

- **توجه:** شما می‌توانید هر تعداد تابع یا متغیر کمکی را در ساختار کامپوننت بازی (یا همان کلاس Game2048) پیاده‌سازی کنید، اما توابع و متغیرهایی که در مورد آن‌ها در سوال توضیح داده شده است نباید تغییری داشته باشند. تغییر در ساختار ورودی توابع یا نام توابع و متغیرهای تعریف شده باعث **عدم دریافت نمره** از سمت داور خودکار خواهد شد.
- **توجه:** تمامی موارد مربوط به استایل‌های بازی در پروژه اولیه از قبل قرار گرفته اند و شما نیازی به پیاده‌سازی هیچ‌گونه استایلی ندارید. در این سوال که از *Tailwindcss* برای طراحی رابط کاربری استفاده شده است، موارد مورد نیاز مربوط به کامپوننت بازی در مسیر `View` این کامپوننت در مسیر `resources/views/livewire/game2048.blade.php` قرار گرفته اند، اما شما نیاز پیدا خواهید کرد تا مانند ساختار گفته شده منطق نمایش آن‌ها را پیاده‌سازی کنید.
- **توجه:** پیاده‌سازی منطق نمایش در ویوی کامپوننت `resources/views/livewire/game2048.blade.php` در سیستم دآوری مورد تست قرار خواهد گرفت، لذا از تغییر نام یا افزودن فواصل کمتر یا بیشتر در متن‌هایمان‌هایی که در اختیار شما قرار داده شده مانند دکمه‌ها و یا متن‌های نمایش داده شده اکیدا خودداری کنید. ایجاد تغییرات منجر به **عدم دریافت نمره** از سیستم دآوری خواهد شد.
- **توجه:** پس از اعمال تغییرات، کل پروژه به غیر از پوشه‌ی `vendor` و `node_modules` را *Zip* کرده و آپلود کنید.
- **توجه:** نام فایل *ZIP* اهمیتی ندارد.