Conception and implementation of protecting the Queen Coding Challenge

1. Conception

To model that problem, I created 3 objects:

* A **point** object: this object represents the coordinates (x, y) of an object in the plan. Point object will then have two attributes to store the abscissa and the ordinate of the point. I also define methods that will operate on the state of that object.
* A **Person** object: It represents the Queen that will be added to the Kingdom
* A **Kingdom** object: It represents the Kingdom with it dimensions where the Queen will be added.

Next I defined a relationship between those objects. A Queen object has a Point object as attribute in order to determine the position of the queen. The object Queen has another attribute to store the direction the Queen is actually facing on. The object Queen belongs to a Kingdom. The kingdom object has methods to move the Queen inside the Kingdom.

1. Implementation

I defined and implemented classes that contain the logic of the challenge. As the development is API-centric I used the Laravel framework to speed up with the implementation of the endpoints of the API. The following lines are the description of the architecture of Laravel and how it works.

The root directory contains a variety of folders:

* The **App** directory contains the core code of the application. Almost all of the classes in the application are in this directory. Inside that directory we have The **Http** directory which contains controllers, middleware, and form requests. Almost all of the logic to handle requests entering the application will be placed in this directory.
* The **Bootstrap** directory contains files that bootstrap the framework and configure auto loading. This directory also houses a cache directory which contains framework generated files for performance optimization such as the route and services cache files.
* The **Config** directory contains all of application's configuration files.
* The **Database** directory contains database migration and seeds.
* The **Public** directory contains the index.php file, which is the entry point for all requests entering the application. This directory also houses assets such as images, JavaScript, and CSS.
* The **Resources** directory contains views as well as raw, un-compiled assets such as LESS, SASS, or JavaScript. This directory also houses all of language files.
* The **Routes** directory contains all of the route definitions for your application. By default, several route files are included with Laravel:  web.php, api.php, console.php and channels.php.
* The web.php file contains routes that the RouteServiceProvider places in the web middleware group, which provides session state, CSRF protection, and cookie encryption. If the application does not offer a stateless, RESTful API, all of your routes will most likely be defined in the web.php file.
* The api.php file contains routes that the RouteServiceProvider places in the api middleware group, which provides rate limiting. These routes are intended to be stateless, so requests entering the application through these routes are intended to be authenticated via tokens and will not have access to session state.
* The console.php file is where we may define all of Closure based console commands. Each Closure is bound to a command instance allowing a simple approach to interacting with each command's IO methods. Even though this file does not define HTTP routes, it defines console based entry points (routes) into your application.
* The channels.php file is where you may register all of the event broadcasting channels that your application supports.
* The **Storage** directory contains compiled Blade templates, file based sessions, file caches, and other files generated by the framework. This directory is segregated into app, framework, and logs directories. The app directory may be used to store any files generated by the application. The framework directory is used to store framework generated files and caches. Finally, the logs directory contains the application's log files.
* The **Tests** directory contains your automated tests.
* The **Vendor** directory contains all Composer dependencies.
* I added the **Documentation** directory with contains all the documentation regarding that challenge.