PROVIDE A GLOBAL POINT OF ACCESS TO A SERVICE WITHOUT COUPLING USERS TO THE CONCRETE CLASS THAT IMPLEMENTS IT.
STRUCTURE

- Service Locator
  - service
  + static Provide(Service)
  + static GetConcreteService()
What happens when the component isn’t there? How do we deal with it?
Option A:

- Check for null and exit early

```csharp
private void OnCollisionEnter()
{
    var score = Services.Score;
    if (score == null)
    {
        return;
    }
    score.AddPoints(hitValue);
}
```
Option B:

- Use NullService

```csharp
private void OnCollisionEnter()
{
    Services.Score.AddPoints(hitValue);
}
```
How can the locator find services?
Option A:

Dependency Injection - outside code registers it

**Pros**
- Very simple
- Can change during gameplay

**Cons**
- Very easy to break
- Locator depends on outside code
- Temporal coupling
Option B:

Set it during compilation

```csharp
public static class Services
{
    #if UNITY_EDITOR
    public static readonly GameScore Score = new LoggedGameScore();
    #else
    public static readonly Score Score = new GameScore();
    #endif
}
```
Option C:

Actually do the locating

- Configuration files
- Reflection

Pros
- Configurable
- Can be reliable

Cons
- Complicated
- Can be slow
<table>
<thead>
<tr>
<th></th>
<th>SINGLETON</th>
<th>SERVICE LOCATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL ACCESS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LIFECYCLE</td>
<td>Single object</td>
<td>-</td>
</tr>
<tr>
<td>NULL REFERENCE</td>
<td>Never</td>
<td>Optional, NullObjects</td>
</tr>
<tr>
<td>UNIT TESTING</td>
<td>Detrimental</td>
<td>Decorators, NullObjects</td>
</tr>
<tr>
<td>INITIALIZATION</td>
<td>Lazy</td>
<td>Dependency injection</td>
</tr>
<tr>
<td>RESPONSIBILITIES</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
TODO

- Read Service Locator chapter from Game Programming patterns and CGLearn
  http://www.gameprogrammingpatterns.com/service-locator.html

- Do the Service Locator task in CGLearn IN TEAMS
  https://cglearn.codelight.eu/student/tasks