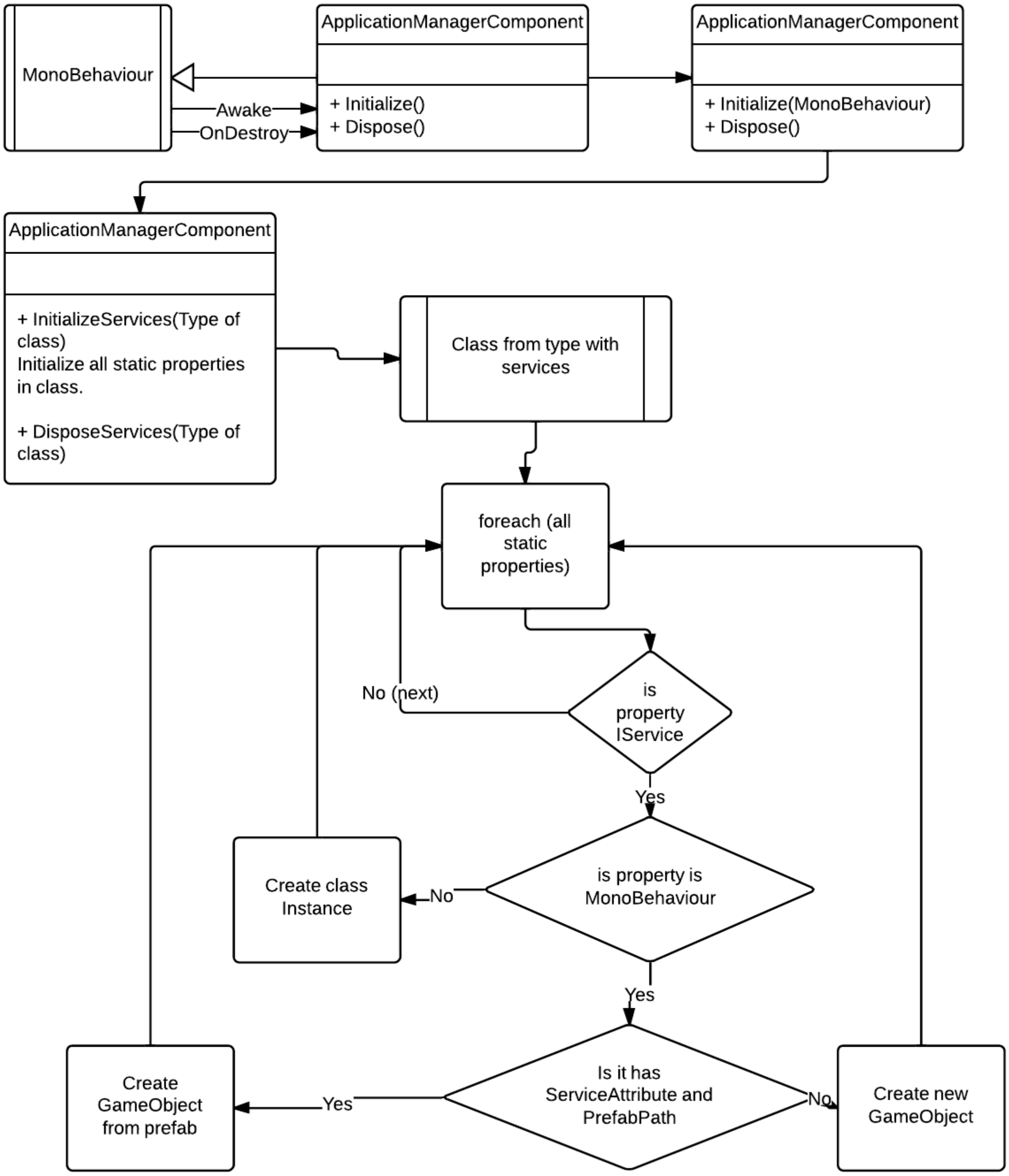
Service structure



MonoBehaviour
Awake
OnDestroy
ApplicationManagerComponent
+ Initialize()
+ Dispose()
ApplicationManagerComponent
+ Initialize(MonoBehaviour)
+ Dispose()
ApplicationManagerComponent
+ InitializeServices(Type of class)
Initialize all static properties in class.
+ DisposeServices(Type of class)
Class from type with services
foreach (all static properties)
No (next)
is property IService
Yes
is property is MonoBehaviour
No
Create class Instance
Yes
Is it has ServiceAttribute and PrefabPath
Yes
Create GameObject from prefab
No
Create new GameObject

ApplicationManagerComponent

- To make only one entry point in our application there is one MonoBehaviour: ApplicationManagerComponent.
- It listens to Awake() and OnDestroy() only to create instance of ApplicationManager

ApplicationManager

- Singleton which initialize when application starts.
- On initialization call ServiceInitializator->InitializeServices for all static service classes
- On dispose call ServiceInitializator->DisposeServices for all static service classes

ServiceInitializator

- Using reflection takes all **public** and **static properties** from class
- Exclude all properties which not using IService interface
- If property is MonoBehaviour then try get ServiceAttribute
    - If there is ServiceAttribute and PrefabPath is not empty -> create GameObject from prefab
    - Else -> create new GameObject
- Else -> create class instance

IService

- Interface with 2 methods: **Initialize** and **Dispose**

ServiceAttribute

- Class attribute with custom parametres:
    - PrefabPath - load gameobject from prefab
    - Dependencies - array of depend Types

Example

```
[Service(PrefabPath = "...somePathToPrefab...")]
public class MonoBehaviourFromPrefab : MonoBehavour, IService
{
        //Load and instantiate gameobjet from prefab
        public void Initialize() { }
        public void Dispose() { }
}

[Service()] //or possible no attribute
public class MonoBehaviourNew : MonoBehavour, IService
{
        //Create new gameobjet and AddComponent<> to it
        public void Initialize() { }
        public void Dispose() { }
}

public class SomeServiceClass : IService
{
        //Create instance of this class
        public void Initialize() { }
        public void Dispose() { }
}

//just simple add classes like properties
public static class _
{
        public MonoBehaviourFromPrefab MonoBehaviourFromPrefab { get; set; }
        public MonoBehaviourNew MonoBehaviourNew { get; set; }
        public SomeServiceClass SomeServiceClass { get; set; }
}
```