Lake Runner

Developers Manual

Environment Setup

Text Editor

Developers can use the text editor of their choice. However, those with support with GitHub may make it easier when coordinating with the other developers.

GitHub

GitHub is used for source control. Each developer will have their own source folder, which they will develop their files in and push to GitHub.

To clone the GitHub repository to your machine, first download the latest version of git from https://git-scm.com/downloads. Then, follow the instructions according to your text editor (if it has integration with GitHub). Otherwise, enter this git command into your terminal.

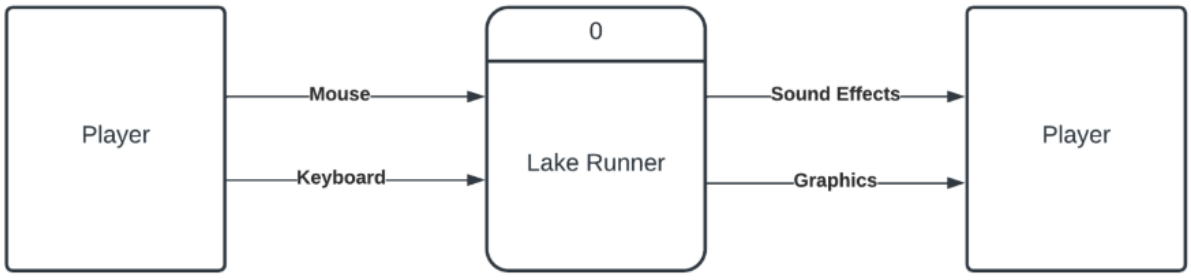
```
git clone git@github.com:ClearLakeStudios/LakeRunner.git
```

Unity

All developers will create their features using the Unity editor version 2021.3.18f1 that can be found at https://unity.com/releases/editor/qa/lts-releases. Launch Unity Hub and press the "Open" button in the upper right corner. Select the cloned repository on your machine.

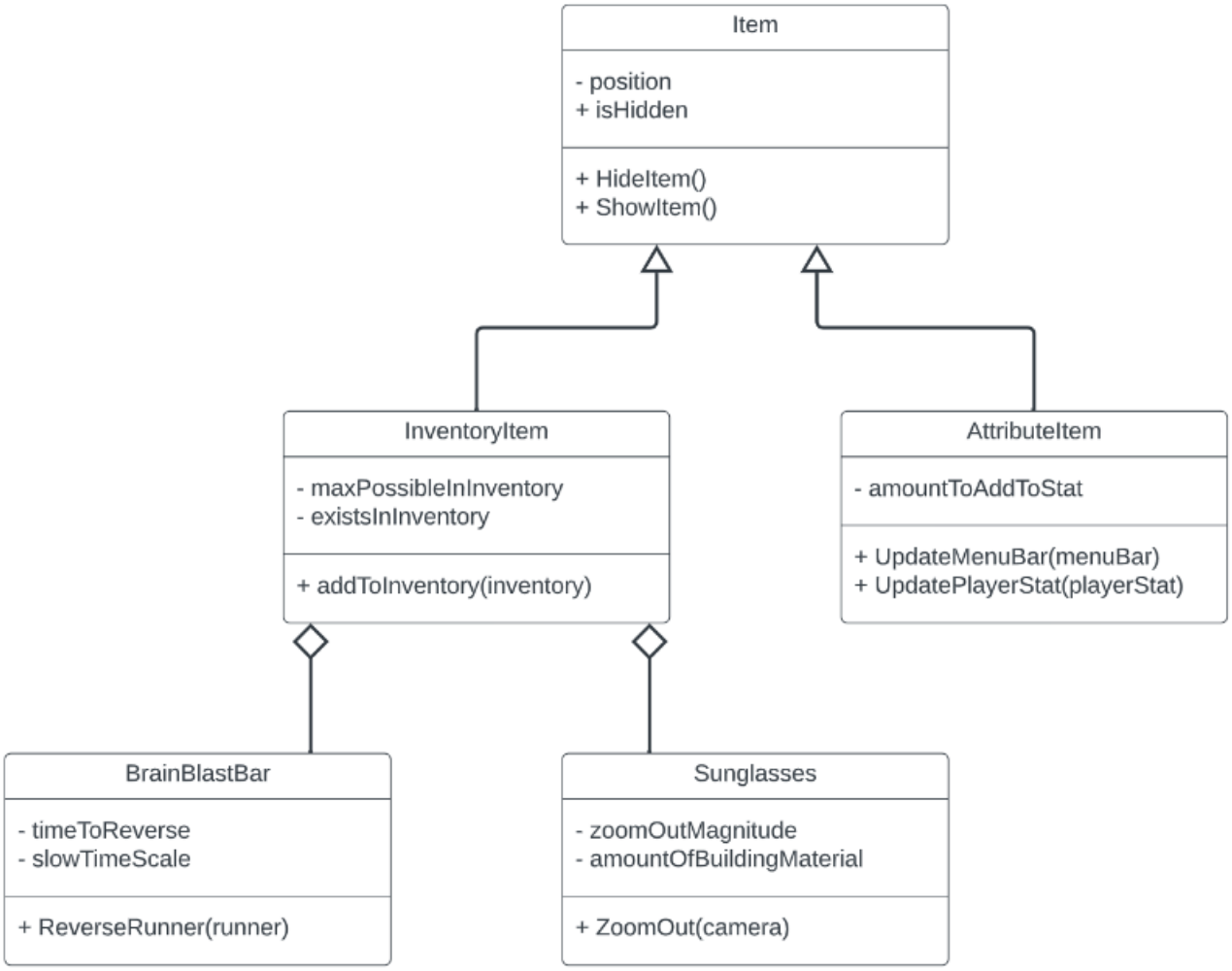
Lake Runner High-Level View

Context Diagram

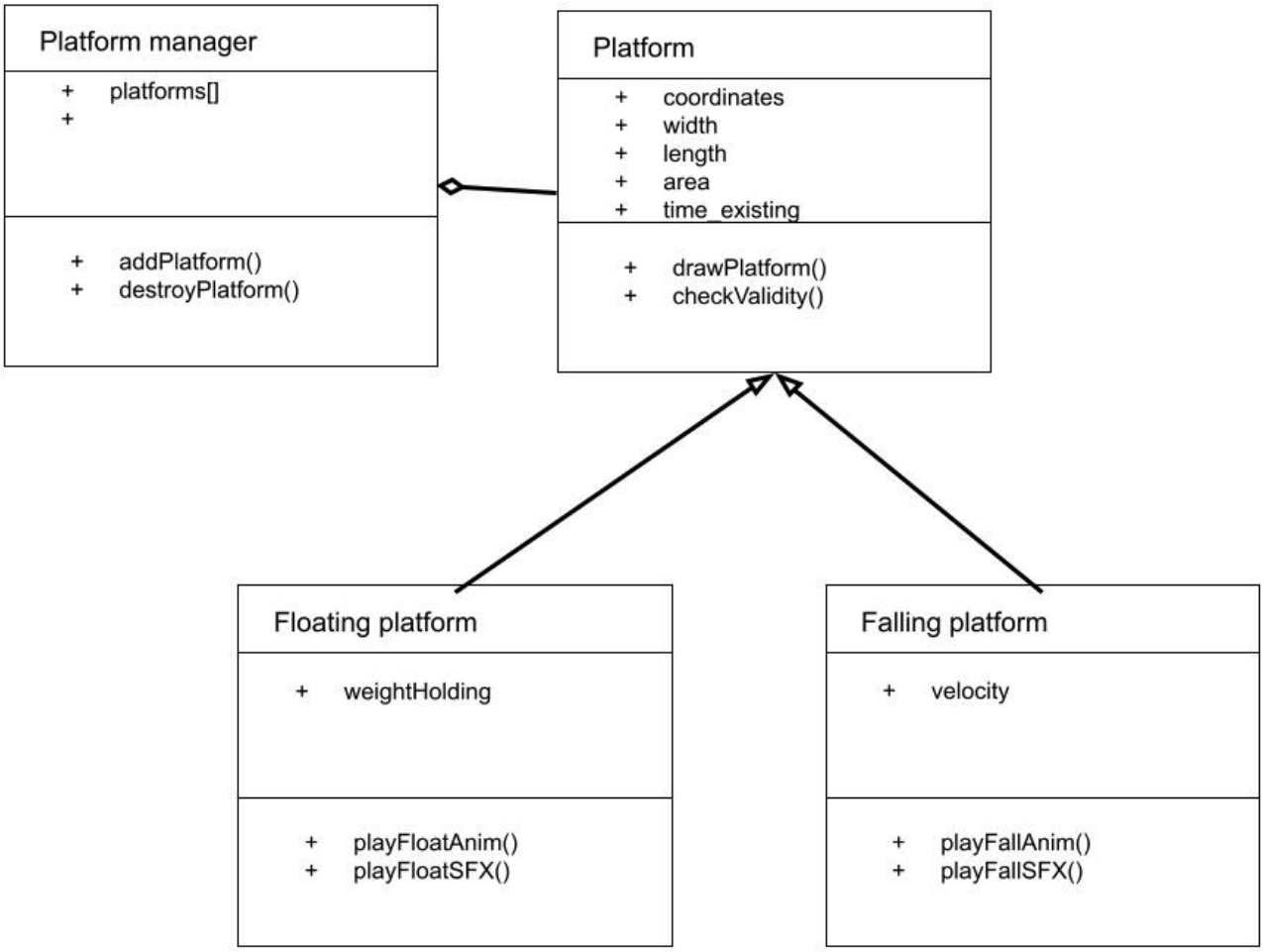


Class Diagrams

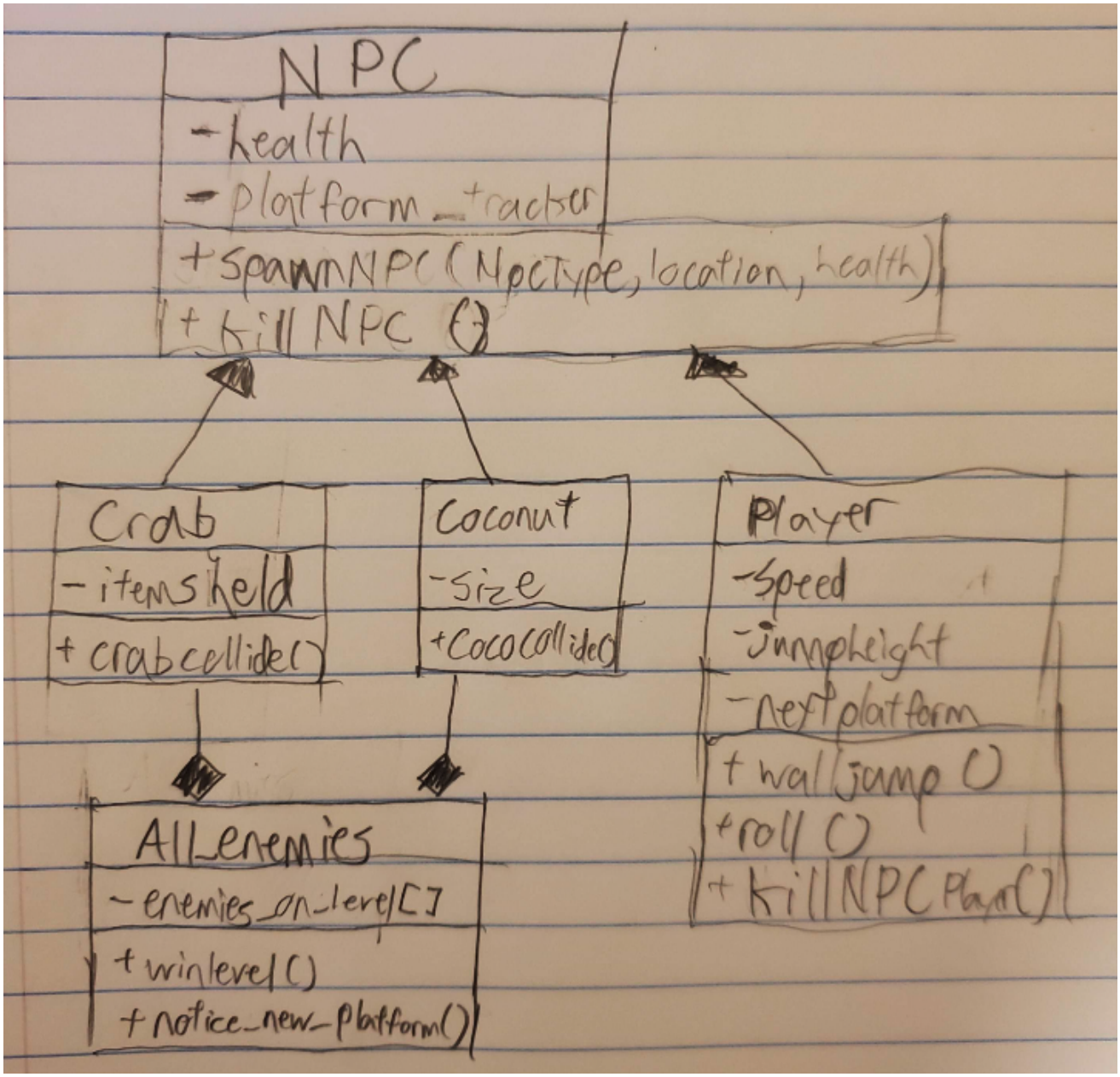
Items & Power Ups



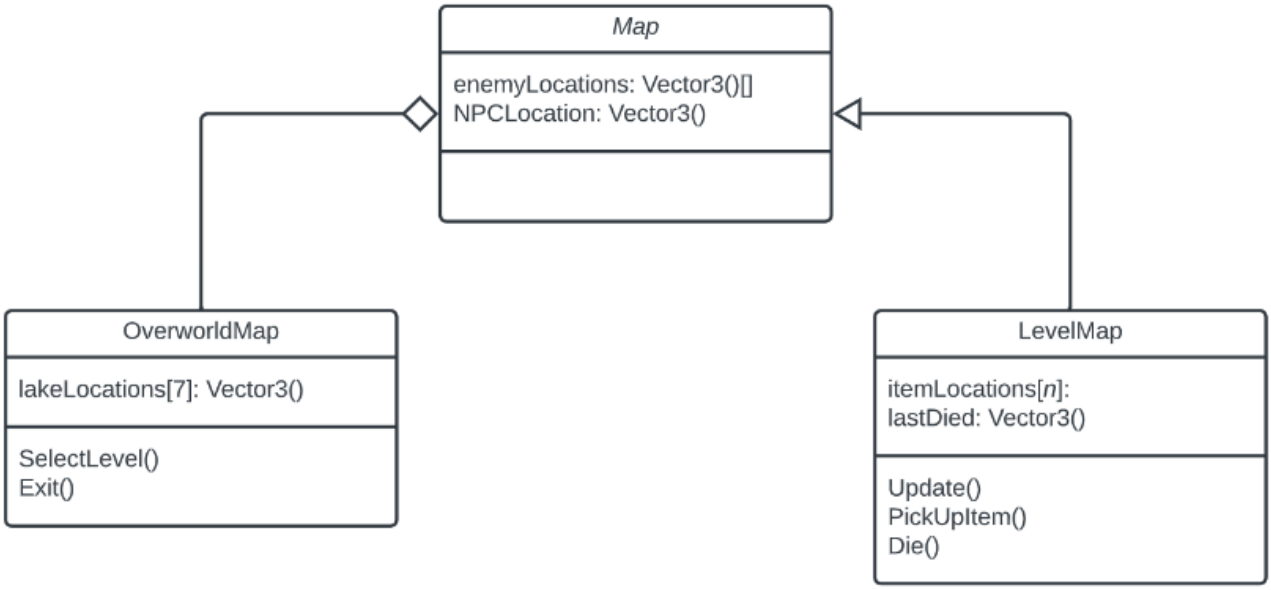
Platforms


Platform manager
+ platforms[]
+
+ addPlatform()
+ destroyPlatform()
Platform
+ coordinates
+ width
+ length
+ area
+ time_existing
+ drawPlatform()
+ checkValidity()
Floating platform
+ weightHolding
+ playFloatAnim()
+ playFloatSFX()
Falling platform
+ velocity
+ playFallAnim()
+ playFallSFX()

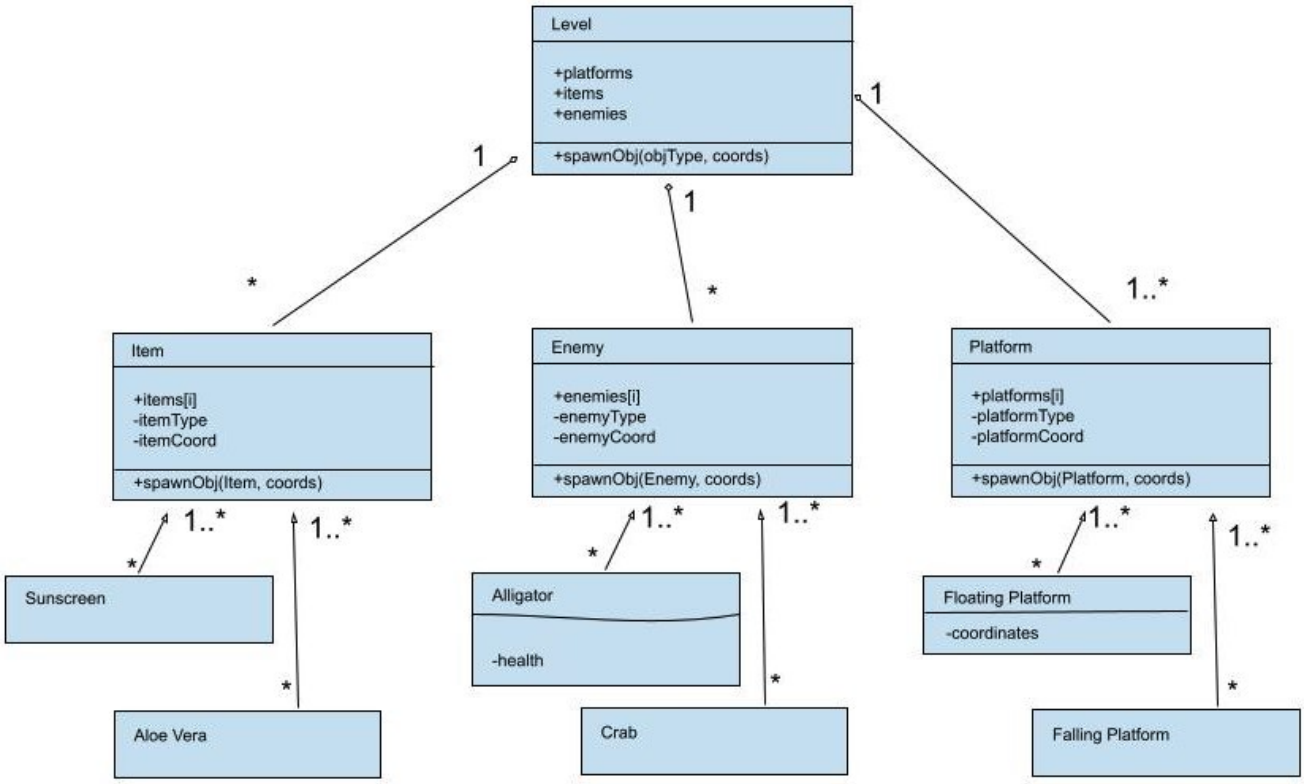
Hero & Enemy NPCs



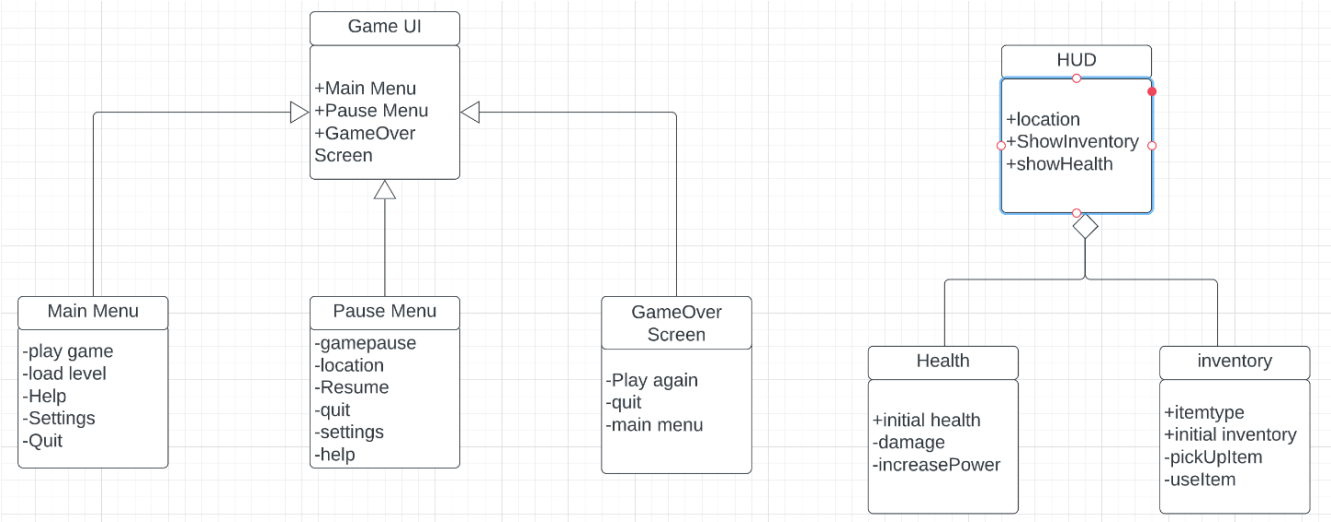
Overworld System & Inter-level Map



Level Generation



User Interface



Oral Exam Technical Topics

Documentation

Follow the documentation guidelines detailed in the **ClearLakeStudios-CodingStandards** document, ensuring comments document all prefabs, C# files, classes, and functions.

Consider the following questions when writing your documentation:

- What question are you trying to answer here?
- Who do you anticipate would be asking that question?

Code Reuse

The code you are developing should be reusable in some way. Write a comment, providing an example of a reuse situation.

Consider the following questions when writing your example:

- What did you have to do to integrate with the code you wrote?
- What are the legal implications if you market your code with the reused portion?

Test Plan

You must design and implement a comprehensive test plan. Provide at least one test case for every non-trivial function.

Consider the following questions when creating each test:

- What are you testing?
- Why did you choose these tests?

Static and Dynamic Binding

You must be able to show one case in your code where there could be either static or dynamic binding.

Mock up some code to show how you would set the static type and dynamic type of a variable.

```
#include <iostream>
using namespace std;

class Grandpa
{
   public:
     virtual void DoDynamic() {cout << "      Dynamic Grandpa" << endl;}
     void DoStatic(){cout << "      Static Grandpa" << endl;}
     void DoPartial() {cout << "      Partial Grandpa" << endl;}

};

class Father: public Grandpa
{
   public:
     void DoDynamic(){ cout << "      Dynamic Father" << endl; }
     void DoStatic(){cout << "      Static Father" << endl;}
     virtual void DoPartial() {cout << "      Partial Father" << endl;}
};

class Son: public Father
{
   public:
     void DoDynamic(){ cout << "      Dynamic Son" << endl; }
     void DoStatic(){cout << "      Static Son" << endl;}
     void DoPartial() {cout << "      Partial Son" << endl;}
};
```

To give you a good idea about static and dynamic binding, refer to the image above.

1. Choose a dynamically bound method. What method gets called now?
2. Change the dynamic type. What method gets called now?

3. Choose a statically bound method. Which method would be called in each of the two previous cases?

Software Patterns

Review the Team Lead 3 presentation on software patterns (located in the "docs/presentation" folder on the GitHub repository). You may also go to https://sourcemaking.com/design_patterns to find a list of software patterns with their class diagrams.

1. Choose and implement at least two patterns in your code.
    a. Read the "problem section" of each pattern.
    b. Determine if the problem is present in your code.
    c. If so, read the "example section" and confirm it is similar to your situation.
    d. Ensure they are relevant and make sense to use.
2. Be able to answer the following about your chosen patterns:
    a. What patterns did you choose?
    b. Why did you choose those patterns?
    c. Is the use of each pattern justified?
3. Choose the pattern you know best (of the two you chose), and be able to answer and do the following:
    a. Would something else have worked as well as or better than this pattern?
    b. When would be a bad time to use this pattern?
    c. Draw the class diagram for it.

Creating a Prefab

Prefabs allow you to store a GameObject object with its components and assets and use it as a reusable asset.

To create a prefab:

1. Select **Asset > Create Prefab** in the Unity Editor.
2. Drag an object from the scene into the "empty" prefab asset was created.
3. To create an instance of a prefab, drag the prefab asset from the project view into a scene view.