1.1

The goals of the system being designed are:

* To create a game that is fun to play and gradually more challenging.
* Designed according to the responsibility-driven-design principles.

After analysing the requirements specification the following noun phrases are collected and separated into: obvious classes, uncertain candidates and nonsense.

|  |  |  |
| --- | --- | --- |
| **Obvious classes** | **Uncertain classes** | **Nonsense** |
| Walls | Levels | Playing field |
| Platforms | Power-ups | 2d-perspective |
| Player | Movement | Fire bubble |
| Enemy | Jump | Die |
| Bubbles | Gravity | Collide |
| Bubble containing object |  | Points |
| Game |  | Time |

Now the noun phrases are divided into conceptual entities and physical objects.

|  |  |
| --- | --- |
| **Conceptual entities** | **Physical objects** |
| Walls | Movement |
| Platforms | Jump |
| Player | Gravity |
| Enemy | Die |
| Bubbles | Points |
| Bubble containing object | Fire |
| Game | Levels |
| Power-ups | Time |

Now values of attributes are:

Movement is x-location and y-location. Same goes for jumping and gravity.

Points is a value of player.

Die is an attribute that can be a value of a player or an enemy in the form of a binary value 1 / 0.

Time is an implicit attribute of the system.

After this analysis the candidate classes are:

|  |  |
| --- | --- |
| **Candidate classes** |  |
| Walls | Game |
| Platforms | Power-ups |
| Player |  |
| Enemy |  |
| Bubble |  |
| FilledBubble |  |

Now it is time to apply the **Class-Responsibility-Collaboration Cards**.

|  |  |
| --- | --- |
| **Wall** | |
| **Superclass(es):** Immutable object | |
| **Subclasses:** | |
| Immutable |  |
| Blocks movement |  |
| Contains level |  |

|  |  |
| --- | --- |
| **Platform** | |
| **Superclass(es):** Immutable object | |
| **Subclasses:** | |
| Immutable |  |
| Can’t fall through |  |

After discussing the classes **Wall** and **Platform** we came to the conclusion that they share the property immutable which is essential for both classes. Thus we introduced an abstract class **Immutable object.** In both cases there are no subclasses since they are both a specific (sub)class of immutable object. At the moment we don’t plan to implement different kinds of walls or platforms.

There are also no collaborating classes we could think of.

|  |  |
| --- | --- |
| **Player** | |
| **Superclass(es):** GravityObject | |
| **Subclasses:** | |
| Gravity | Platform |
| movement | Wall, FilledBubble |
| Fire bubble | Bubble |

|  |  |
| --- | --- |
| **Enemy** | |
| **Superclass(es):** GravityObject | |
| **Subclasses:** | |
| Gravity | Platform |
| movement | Wall, Bubble |

After considering that both Player and Enemy are subject to gravity. We decided to introduce the abstract class GravityObject. Furthermore the Player can fire bubbles and interact with walls, FilledBubbles and Platforms. The Enemy interacts with the Platforms, wall and bubbles.

|  |  |
| --- | --- |
| **Bubble** | |
| **Superclass(es):** Floating object | |
| **Subclasses:** | |
| Floats |  |
| Catch an enemy | Enemy |

|  |  |
| --- | --- |
| **FilledBubble** | |
| **Superclass(es):** Floating object | |
| **Subclasses:** | |
| Floats |  |
| Collides with the player | Player |

After considering that both Bubble and FilledBubble have the special property that they float and move in the +-y direction. We introduced the abstract class floating object.

|  |  |
| --- | --- |
| **Game** | |
| **Superclass(es):** Floting object | |
| **Subclasses:** | |
| Retrieves the information and displays it. | All other classes |

Now it is time to discuss the **Responsibilities** of all the objects.

The responsibilities of a **wall** object are:

* Stops other objects from moving through it
* Contain the level

The responsibilities of a **platform** object are:

* If an object is on the platform, they are not subject to gravity

The responsibilities of a **player** object are:

* Can move in the level
* Can die if it collides with an enemy
* Can shoot bubbles
* Can kill an enemy by colliding with a filled bubble

The responsibilities of an **enemy** object are:

* Kill the player
* Can move in the level
* Can be contained by a bubble

The responsibilities of a **bubble** object are:

* Float in the level
* Contain an enemy
* Disperse after a set amount of time

The responsibilities of a **filledbubble** object are:

* Float in the level
* Disperse after a set amount of time which releases the contained enemy

The responsibilities of the **game** class are:

* Gather all information and display it
* Provide means to pause and continue the game

Now we can compare this design to the actual implementation.

In our implementation we made correct use of polymorphism and abstract classes. We identified the classes which have a “kind-of” hierarchy. However in our implementation we found out that a game-class is not sufficient. The framework we picked allowed us to make better use of MVC. We implemented one main class game, which gets the views from the logic controller class.