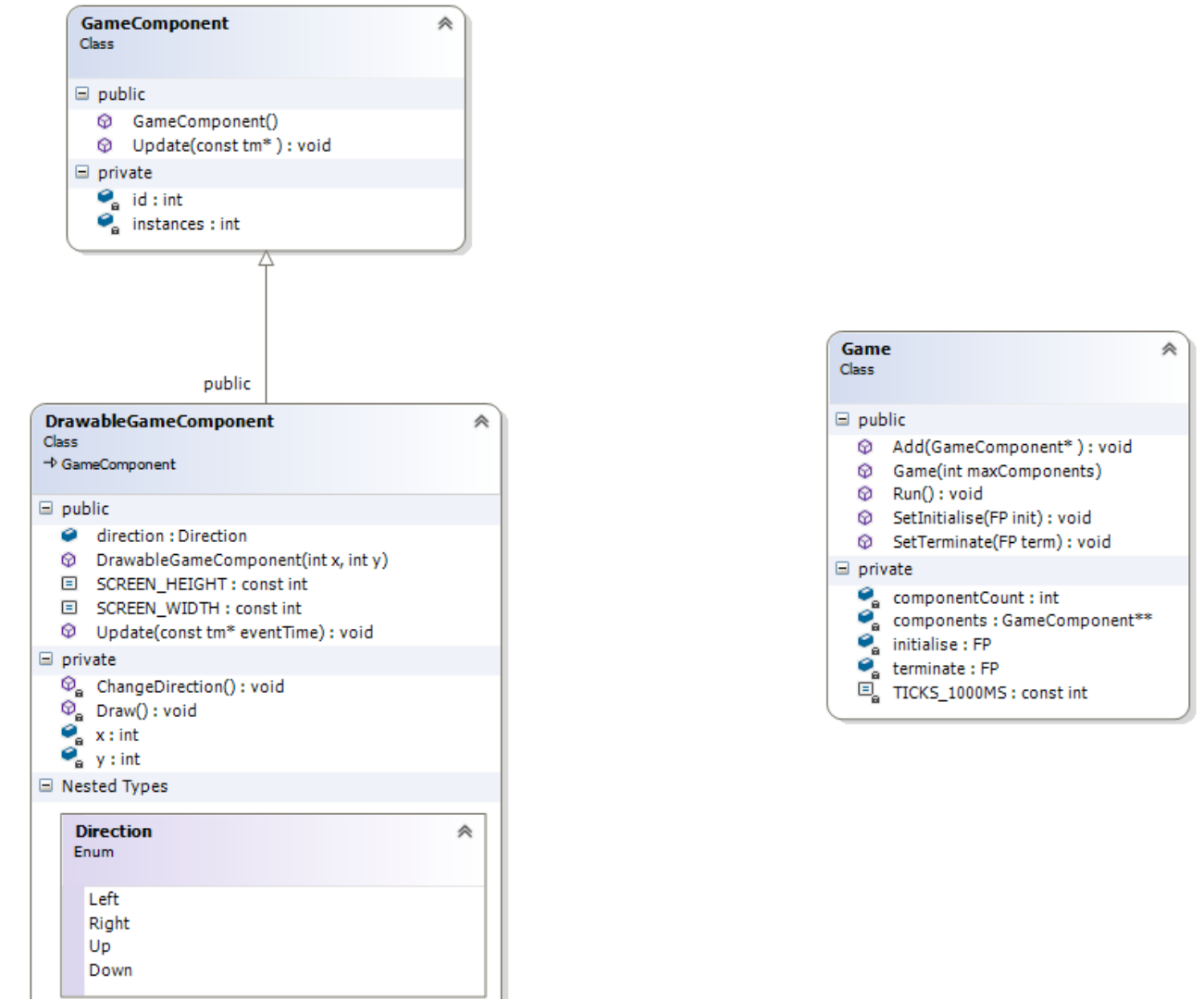


Task 1

The classes below implement a simple game engine that consists of two classes representing entities within the game (GameComponent & DrawableGameComponent, which has an associated position – x,y) and a controlling class (Game) that manages the entities.



Game

Add : adds a new (GameComponent/DrawableGameComponent) object to an array of pointers of type GameComponent (components).

componentCount: Number components in array.

initialise: Pointer to standalone function

terminate : Pointer to standalone function.

SetInitialise : assigns the address of a standalone function to the Initialise data member.

SetTerminate : assigns the address of a standalone function to the terminate data member.

FP is a typedef representing a type of function that returns void and has no parameters.

Run : Invokes the functions whose address has been assigned (using SetInitialise) to the Initialise pointer . This standalone function should just display the words “Initialising game” within the console. Run implements a loop which iterates through the components array invoking the component’s Update member functions. The time of invocation is passed as an argument to the Update member function. The component’s Update member function should be invoked **once every second**. But should only execute 5 times.

Finally the function whose address has been assigned (using SetTerminate) to the terminate pointer should be invoked. This function should just display the words “terminating game” within the console.

GameComponent.

Represents non visual entities within a game.

id : Each entity has a unique ID number. This ID is generated automatically using the instances static member, which records a running total of the number of instances instantiated from either the GameComponent or DrawableGameComponent classes. Therefore the first object will have an id of 1, the second 2 etc..

Update: Display the objects id and time at which the Update member function was invoked.

DrawableGameComponent

Represents visible 2D game entities.

Constructor:

Set the x and y values to zero and direction to Right

Update:

1. Displays the object's id and time of invocation.
2. Increments / decrements the x or y values depending upon the current direction (Up, Down, Left, Right) .
3. Invokes the Draw() member function
4. Invokes the ChangeDirection member function.

Note: The x & y values should not exceed the screen size. Valid values are

- x in the range 0 .. 80
- y in the range 0..20

x and y values should be clamped if necessary. E.g. if y = -1 set it to 0.

Draw : Displays the current direction along with the x and y values on the same line.

ChangeDirection : assigns a new random direction to the direction data member. This direction must be different to the current direction.

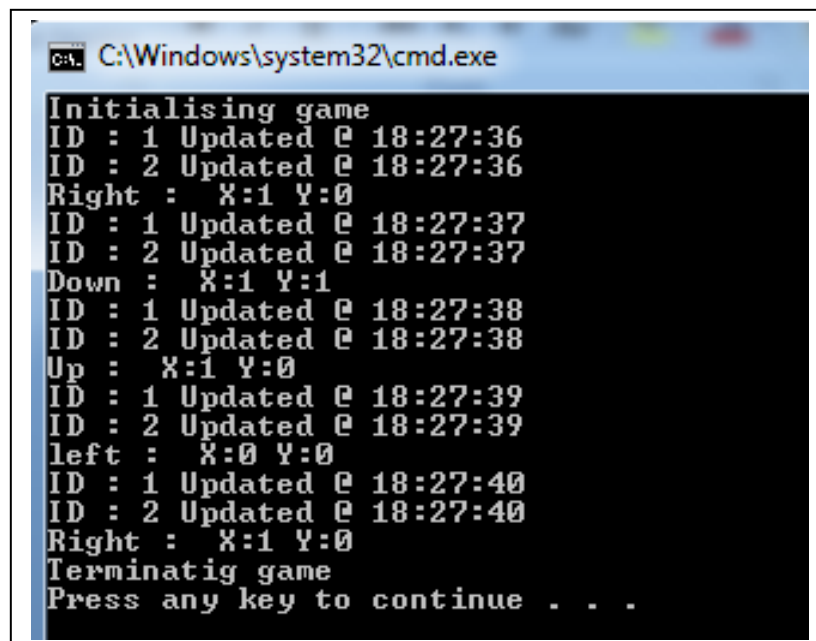
Instructions

Implement the classes and define the two standalone functions above main.

Within main -

1. Create a dynamic instance of Game
2. Invoke SetInitialise passing it one of the standalone function addresses.
3. Invoke SetTerminate passing it the other standalone function address.
4. Add a GameComponent object with an id of 1 to the game object
5. Add a DrawableGameComponent object with an id of 2 and a position of x=0, y=0 to the game object.
6. Invoke the game object's run member function.

Your output should be similar to that below. Although clearly the random directions may be different.



```
C:\Windows\system32\cmd.exe
Initialising game
ID : 1 Updated @ 18:27:36
ID : 2 Updated @ 18:27:36
Right : X:1 Y:0
ID : 1 Updated @ 18:27:37
ID : 2 Updated @ 18:27:37
Down : X:1 Y:1
ID : 1 Updated @ 18:27:38
ID : 2 Updated @ 18:27:38
Up : X:1 Y:0
ID : 1 Updated @ 18:27:39
ID : 2 Updated @ 18:27:39
left : X:0 Y:0
ID : 1 Updated @ 18:27:40
ID : 2 Updated @ 18:27:40
Right : X:1 Y:0
Terminatig game
Press any key to continue . . .
```