General Notes

- Comment structure needs to be consistent across project. (Decide how to comment methods vs in-methods, requires clauses, etc.)
- We need to decide on namespaces and project file structure.
- Decide if we want author tags on classes.
- Die and Dead method names used interchangeably, should probably pick one.
- If classes have methods that contains interactables and are not implemented for this sprint, please comment this in the code! There is a lot of unimplemented code with no comments.
- FireballUpdate class must be added to Zenhub.
- Booleans default to false in C# so all boolean fields that you want as false you can leave as just private bool myVar;
- Need to be consistent with naming conventions, some fields are capitalized, others are not
- In general our Update methods are overly long and complex
- Getting rid of magic numbers

- **Interfaces**
  - ICollectable
    - N/A
  - ICommand
    - N/A
  - IController
    - Needs comment explaining parameters, unclear why there is both Rectangle and Key.
  - IDrawable
    - Comment for “IsDrawn” should be ‘Reports whether <c> this </c> is drawn to the screen or not’?
    - For the “ToggleDrawing” function, what/where is it toggling?
  - IEnemy
    - State and draw interface comment not necessary.
    - Change direction method comment unclear.
    - Fall method comment unclear, falling into bottom is vague to me.
    - GetPosition method comment I think over explains the method and makes the method less general than it could be
  - IItem
    - N/A
  - IMoveStateMachine
    - Fields may want to explain an example initial state, to me it is a little confusing why the dictionary holds a string and a bool.
    - Who will be setting all states?
  - IPlayer
    - Dead code left in an line 39, do we need UpdateScore in IPlayer?
  - ISprite

        - N/A
    - IStructure
        - Probably needs to implement ICollideable eventually, would there ever be a structure that can't be collided with?
    - IStructureStateMachine
        - Fields may want to explain an example initial state, to me it is a little confusing why the dictionary holds a string and a bool.
        - Who will be setting all states?
    - IThinkingStateMachine
        - N/A
    - IUpdateable
        - N/A
- **Classes**
    - BackgroundObject
        - Background sprite comment at line 7 seems unnecessary.
        - I don't think it needs to implement IItem.
        - If the object gets an ISprite object, then it shouldn't need Columns, Rows, SourceRectangle, or Texture variables
        - I don't think this needs Mario's position.
        - What is graphics variable used for? It is never used.
        - Right now the Draw method is commented out and the 2 other methods don't do anything, so I'm not sure what this class is for.
        - If implementing draw method, should implement IDrawable.
        - Constructor takes too many parameters.
    - BigMarioDecorator
        - In "Update" function, instead of checking for StateMachine.states["Shrunk"], check for "Damaged" and StateMachine.health < 2. And call StateMachine.Shrunk().
        - I think that the MarioDecorator class in the same file as BigMarioDecorator is just supposed to be the Mario class that is in it's own separate file already (Look at the Damaged Link decorator example on Kirby's website)
    - Block
        - Needs comments on why specific methods aren't implemented (Interact, Collision)
        - Would it make more sense to send the sprite in with the constructor, as a block will always have a sprite? Or does it already have a default sprite?
    - BlockStateMachine
        - Why is the block's initial position always 400?
        - Line 20 may create an alias of states and may not save the actual states correctly if states gets changed.
        - Confusing why Update is included then not implemented.
    - BlockSwitchingCommand

        - There should be 1 command class for switching items (combine block, item, and enemy switching)
    - EnemySwitchingCommand
        - See BlockSwitchingCommand
    - FireballStateMachine
        - Line 22 may create an alias of states and may not save the actual states correctly if states gets changed.
        - A switch case may be cleaner looking in Update method.
    - FireballUpdate
        - Should not need a Sprite variable. The fireball class will hold this.
        - Class name is ProjectileUpdate but is called FireballUpdate on Github?
    - FireMarioDecorator
        - In “Update” function, instead of checking for StateMachine.states[“LoseFire”], check for “Damaged” and StateMachine.health < 3. And call StateMachine.LoseFire().
        - I think that the MarioDecorator class in the same file as FireMarioDecorator is just supposed to be the Mario class that is in it’s own separate file already (Look at the Damaged Link decorator example on Kirby’s website)
        - Extra note, just to make sure we know, when Mario takes damage with Fire power, he still remains Big Mario. So the decorators should be layered in a way that looks like FireFlower(BigMario(SmallMario())) and have to remove one at a time
        - Class needs comments.
    - Flag
        - Include comments on why certain things aren’t implemented.
        - In HasBeenInteracted and IsBeingInteracted, “ret” isn’t the most descriptive variable name.
        - Needs comments.
    - FlagStateMachine
        - Why is flags initial position 400?
        - Needs comments on why update is not implemented.

    - GameObjectManager
        - May need a backgroundList when we get to background objects.
        - We do not need a Piranha enemy for level 1 as far as I know
        - The StateMachines for most of the Objects should have a Reset() function that resets the initial position already, as well as their states. They also have their very initial positions hardcoded in there. Do we want the GameObjectManager to send the initial conditions to the statemachines for this sprint?
        - When creating the objects, we need to make sure their constructors get passed the correct things they need.

        - Do the fields of this class need to be public?
        - In lines 175-181, I think this would not work for multiplayer
    - GameState
        - Very long, may want to consider breaking this up a bit in some way if possible.
        - Should there be a GameState interface? Right now a lot of our classes don't have method comments because of the interfaces.
        - Formatting is a little off, spacing is not consistent.
        - Some of these methods can and should probably be private if only this class is calling them
        - PrevWorld should probably have a requirement that level is not 1.
    - Goomba
        - Should probably have variables for 1,2,3, and 5 so they are not magic numbers.
    - GoombaStateMachine
        - Update seems a little long and overly complex.
    - GoombaUpdate
        - Update seems a little long and overly complex.
    - ItemStateMachine
        - Why is initial position 100,300?
        - Position has 2 values at construction but only 1 at reset.
        - Switch case may be cleaner for Update.
    - ItemSwitchingCommand
        - See BlockSwitchingCommand
    - ItemUpdate
        - It's not possible for hitBlock or collected to ever be updated here so those conditions will never trigger.
    - KeyboardController
        - N/A
    - Koopa
        - -2,-1,5,1,3 are magic numbers
    - KoopaStateMachine
        - Update method is overly long and complex
        - Spacing is inconsistent
    - KoopaUpdate
        - N/A
    - LittleMarioDecorator
        - In "Update" function, instead of removing the decorator, it should be adding the BigMario decorator if Mario has grown. And knowing that Mario has grown isn't the StateMachine's job, the StateMachine should actually be getting told about Mario growing itself.
        - The MarioDecorator class issue is the same as the other decorators
        -
    - Mario

        - Mario only needs 1 StateMachine, there are 2 that are instantiated though
        - “Jump” function should call StateMachine.Jumping(GameTime gameTime) instead of “InAir” and send that function gameTime
        - “TakeDamage” function should send StateMachine.Damaged() gameTime
        - If GameObjectManger is calling Mario.Update() then Mario is going to have to have an update function, even if that means it’s just calling statemachine.Update(gameTime);
        - Dead code left in, is update needed?
    - MarioJumpCommand
        - If the space bar is held down, will that continue to execute this command continuously? Or will it activate only once? This affects the statemachine and how it updates Mario’s position on screen.
    - MarioLeftCommand
        - See MarioJumpCommand
    - MarioRightCommand
        - See MarioJumpCommand
    - MarioStateMachine
        - 8,-8, 1.25, 5, 0.5 are magic numbers
        - Update is very long
        - Dead code left in ChangeScore(?)
    - MarioThrowingCommand
        - See MarioJumpCommand
    - MouseController
        - N/A
    - Pipe
        - Need to comment why specific things aren’t implemented
        - Need to move to Review in ZenHub
        - Do fields need to be public?
    - PipeStateMachine
        - 400 is magic number.
        - Need to comment why Update is not implemented.
    - QuitGameCommand
        - N/A
    - Shell
        - The speed shouldn’t be needed here. The statemachine will handle that, so it should be calling statemachine functions instead of setting values on its own for a speed inside itself.
        - Needs to be moved to review in zenhub.
        - -1 and 1 and -2 are magic numbers
    - ShellStateMachine
        - Update may be cleaner as switch case.
        - -3, 3, 1, -1 are magic numbers
    - ShellUpdate
        - N/A

- Sprite
  - Is it at all possible to combine the 2nd and 3rd constructors into 1 with the same parameters as the 2nd? And to hold up what the 3rd constructor does we would just have to send in SpriteEffects.None into that last parameter.
  - It makes more sense to check for IsDrawn in the Game Object Manager before you try to draw the sprite
  - Make sure to move over update logic to game object manager
- SpriteFactory
  - Quite a long class, is there any way we could simplify/shorten it?
  - I would think changing “HitBlock” to something like “UsedBlock”, “FinishedBlock”, or “DoneBlock” would help with clarity.
  - All Texture2D fields could be compiled into one line so it doesnt take up so much space
    - Ex. private Texture2D brickSheet, uBlockSheet, pipeSheet etc.
  - Make sure to do the “TODO”s and “FIXME”s
- StarMarioDecorator
  - In the “Update” function, losing the star depends on time. And the decorator should be tracking this. Then when time runs out, it call StateMachine.LoseStar() so the statemachine knows Mario no longer has the star.
  - MarioDecorator class issue is the same as the above decorators’ issues.
- Text
  - I don’t think field variable names should be capitalized.
- ResetCommand
  - N/A