General notes:

- Should merge into one to run, which help us find more questions
- duplicate operation in some functions
- long line for somes classes
- replace magic numbers
- decide author tags
- should have a folder to contain each category classes (namespace problem)
- Need to be consistent with naming conventions, some fields are capitalized, others are not
- May need a draw method for each related classes
- to many public declaration, change to private declaration

- **Interfaces**
    - ICollectable
        - two bool methods seem to overlap with CollectionState method.
    - ICollideable
        - SLIPING TO NEXT SPRINT, But I will prefer return bool
    - ICommand
        - N/A
    - IController
        - No idea what RegisterCommand method use for
    - IDrawable
        - N/A
    - IEnemy
        - may need to pass parameter (GameTime)for statemachine
    - IInteractable
        - N/A
    - IItem
        - N/A
    - IMoveStateMachine
        - N/A
    - IPlayer
        - may need a draw method()
    - IProjectile
        - N/A
    - ISprite
        - N/A
    - IStructure
        - N/A
    - IStructureStateMachine
        - Break and Destroy Methods seem overlap
    - IThinkingStateMachine
        - N/A
    - IUpdateable
        - Need contain parameter like GameTime
- **Classes**
    - BackgroundObject

        - Don’t need implement IItem
        - Don’t need MarioPosition because Mario should always in the middle of screen
        - Don’t need columns and rows which are done by sprite.
        - In general, too many parameter unnecessary
    - 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
        - may need a more detailed class for each situation
    - BlockStateMachine
        - It shouldn’t have an initial position which should pass from outside like the sprite method.
    - BlockSwitchingCommand
        - Switch Command are similar and looks well
    - EnemySwitchingCommand
        - Switch Command are similar and looks well
    - FireballStateMachine
        - N/A
    - FireballUpdate
        - N/A
    - 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
    - Flag
        - N/A
    - FlagStateMachine
        - Not sure whether the Flag can be broke
    - 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.
    - GameState
        - Too long for checking.
    - Goomba
        - position overlap with state machine update()
        - Magic numbers
    - GoombaStateMachine
        - Using FaceLeft as a direction is enough. !FaceLeft for the right side
    - GoombaUpdate
        - Make it more simple and clear
    - ItemStateMachine
        - N/A
    - ItemSwitchingCommand
        - Switch Command are similar and looks well
    - ItemUpdate
        - Make it more simple and clear
    - KeyboardController
        - N/A
    - Koopa
        - position overlap with state machine update()
        - Magic numbers
    - KoopaStateMachine
        - Using FaceLeft as a direction is enough. !FaceLeft for the right side
    - KoopaUpdate
        - Make it more simple and clear
    - 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
        - duplicate StateMachine Declarations
        - should call StateMachine.Jumping(GameTime gameTime) instead of “InAir”
        - I consider “StateMachine.Update()” is 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

        - N/A
    - MarioThrowingCommand
        - See MarioJumpCommand
    - MouseController
        - N/A
    - Pipe
        - N/A
    - PipeStateMachine
        - Not sure whether the Pipe can be broke
    - QuitGameCommand
        - N/A
    - Shell
        - Not sure what interaction will happen,
    - ShellStateMachine
        - Not sure whether the Shell can be broke
    - ShellUpdate
        - Make it more simple and clear
    - Sprite
        - Actually I prefer to create different classes for different items, mario, enemy instead of getting into the whole one.
        - It makes more sense to check for IsDrawn in the Game Object Manager before you try to draw the sprite
    - 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
        - N/A
    - ResetCommand
        - N/A