A discussion on how your design for assignment 1 helped or hindered your extensions made in this assignment

Change:

The Floor has been changed from consecutive instances of platform to one instance of platform with width of levelWidth, because of performance issues.

JSONReader has now be renamed to JSONInterpreter and be moved to model package. As the name suggest, it is now responsible to not only reading JSON file, but also using EntityFactory to initialize the Level and assign Entity(s) to the Level.

Kept:

Level is still responsible for how the ballboy move.

GameEngine still handle all the entity interaction and instruct the entity in the currentLevel to progress.

Entitys are still split into MovableEntity and StillEntity, this help reducing execution time in LevelImpl.tick(). Tick() determine the interaction between Entitys, and this segregation introduce a first round of type check, so each iteration can run the first round of type check before type check for each type. For instance, only MovableEntity should invoke move() on each iteration.

Rationalise changes you have made to your assignment 1 design

BlockedBackground will now take the config as JSON states to enable different appearance for level, as the specification required.

To enable the ballboy “continuously bouncing”, Enemy and Ballboy now will be affected by gravity.

Ballboy now hold a few more attributes include

- The values to enable the "ballboy continuously bouncing and has some control over the height of the bounce"
    - landed: if landed is true, Ballboy now will bounce() on each tick(). Landed will be turned to true every time collide with instances of Platform. Landed will be turned to false on invoke of bounce().
    - bounceForce: initial value 2.0, and minimum value 2.0. It will increase by 1.0 if UP key is pressed. It will decrease by 1.0 if DOWN key is pressed.
- The values to enable the "ballboy can accelerate left and right (until top speed is reached)"
    - MoveSpeed: the absolute value of moving speed, it will continuously increase LEFT or RIGHT key is pressed, capped at 4.0.
    - toRight: boolean value to indicate whether to accelerate to right.
    - toLeft:boolean value to indicate whether to accelerate to left.

Ballboy now hold a few more methods include

- setter method for toRight, toLeft and landed, getter method for landed.
- boostHeight() and dropHeight() for change bounceForce.
- accelerateToRight() and accelerateToLeft() to tweak moveSpeed and xVel.
- stop() to set toLeft and toRight to false, and moveSpeed and xVel to 0.0.

Factory method pattern

Where you used it

ballboy.factory.EntityFactory is the *Creator*.

ballboy.factory.EntityFactory.create() is the *creator methods*.

ballboy.model.Entity is the *product interface*.

ballboy.model.Ballboy , ballboy.model.Cloud , ballboy.model.Enemy , ballboy.model.Finish , ballboy.model.Platform are the *Concrete Product*.

Ballbyboy.model.JSONInterpreter is the *context*.

What this pattern does for your code in terms of SOLID/GRASP principles

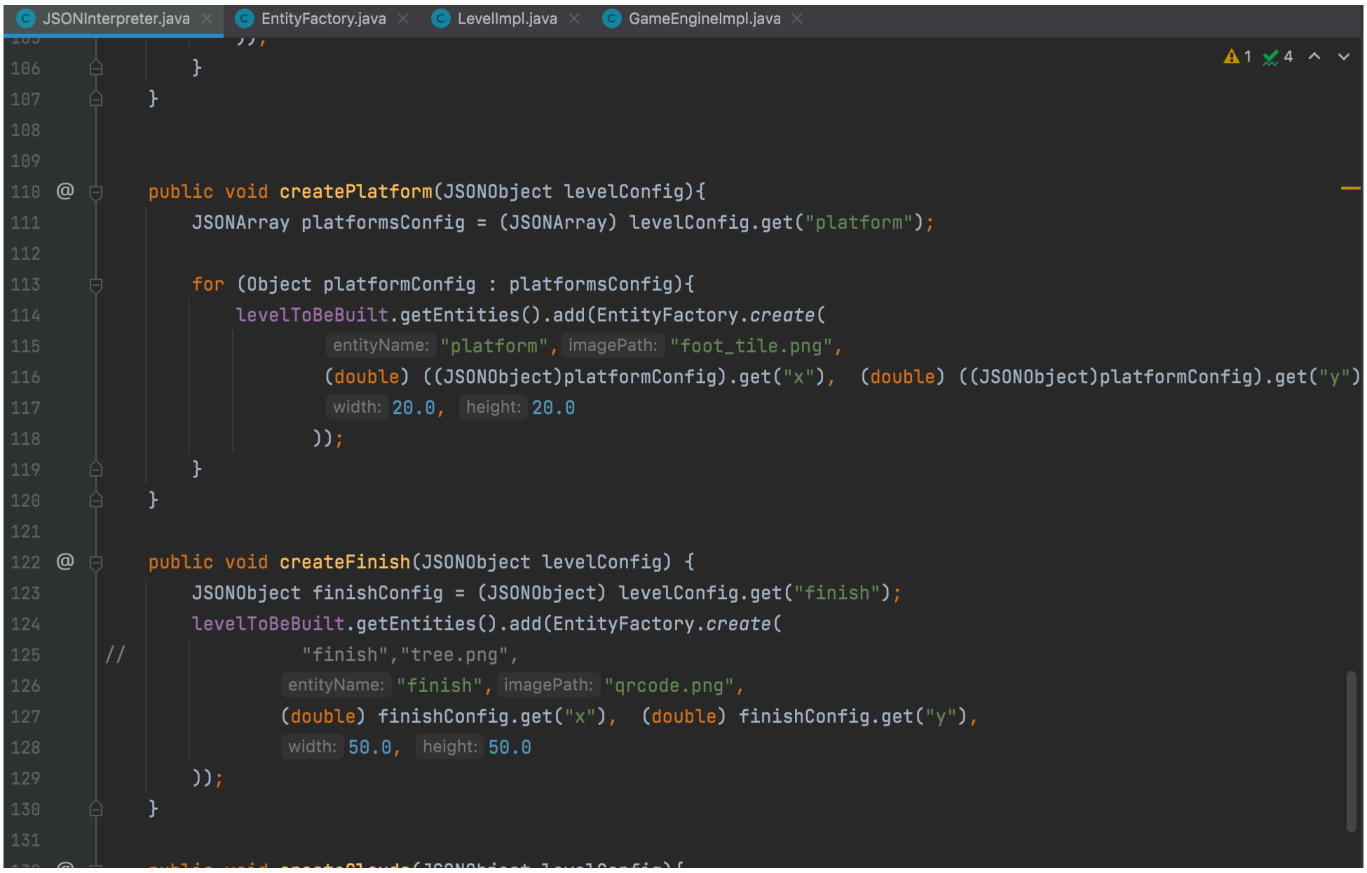
Entityfactory take the responsibility of creation of all the Entity, this maintain the *Single Responsibility Principle*.

Also, new Entity can be introduced without breaking the code in the *context*, and this maintain *Open/Closed Principle*.

Low coupling can be maintained by separating the *creator* and the *concrete products*.

What overall benefits this pattern provides (be specific to your code, not the pattern in general)

EntityFactory has three static method that are called create(), but with different signatures. Thus in JSONInterpreter, without breaking the existing code, new Entity can be introduced and loaded by just invoke the existing create() or new create() if parameters differ.



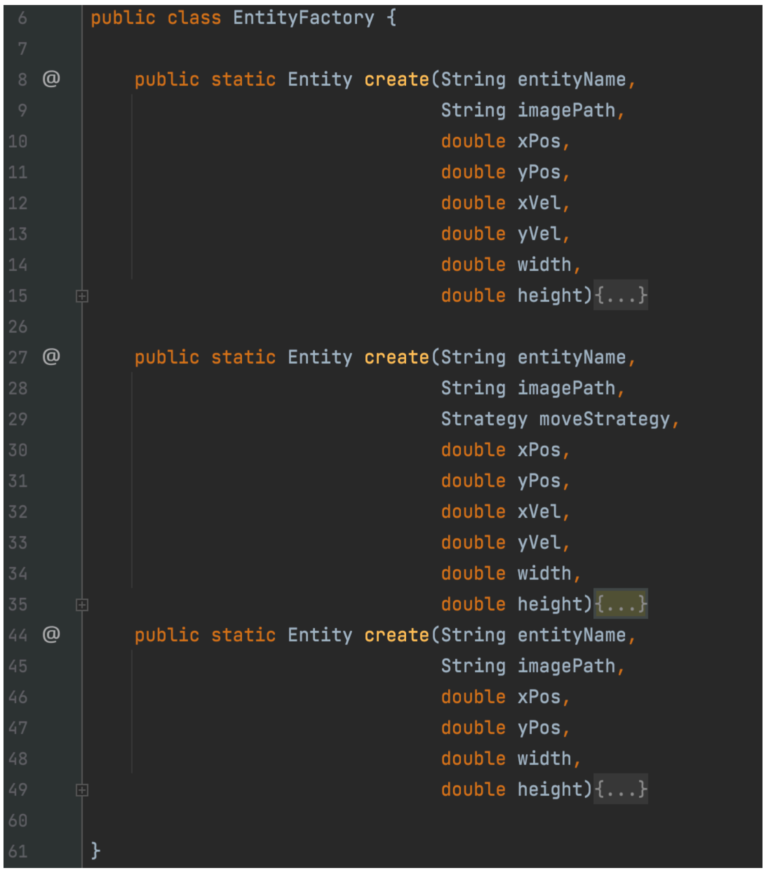
```java
            ));
        }
    }


    public void createPlatform(JSONObject levelConfig){
        JSONArray platformsConfig = (JSONArray) levelConfig.get("platform");

        for (Object platformConfig : platformsConfig){
            levelToBeBuilt.getEntities().add(EntityFactory.create(
                    entityName: "platform", imagePath: "foot_tile.png",
                    (double) ((JSONObject)platformConfig).get("x"),  (double) ((JSONObject)platformConfig).get("y")
                    width: 20.0,  height: 20.0
                   ));
        }
    }

    public void createFinish(JSONObject levelConfig) {
        JSONObject finishConfig = (JSONObject) levelConfig.get("finish");
        levelToBeBuilt.getEntities().add(EntityFactory.create(
//                 "finish","tree.png",
                entityName: "finish", imagePath: "qrcode.png",
                (double) finishConfig.get("x"),  (double) finishConfig.get("y"),
                width: 50.0,  height: 50.0
        ));
    }

    public void createClouds(JSONObject levelConfig){
```



```java
public class EntityFactory {

    public static Entity create(String entityName,
                                String imagePath,
                                double xPos,
                                double yPos,
                                double xVel,
                                double yVel,
                                double width,
                                double height){...}

    public static Entity create(String entityName,
                                String imagePath,
                                Strategy moveStrategy,
                                double xPos,
                                double yPos,
                                double xVel,
                                double yVel,
                                double width,
                                double height){...}
    public static Entity create(String entityName,
                                String imagePath,
                                double xPos,
                                double yPos,
                                double width,
                                double height){...}

}
```

What drawbacks this pattern causes

The code might be overcomplicated since introduce a new class(EntityFactory) to implement the pattern, if the amount of Entity is not huge. But this can be passed on for a better structure and maintaining the SOLID principle and the GRASP principle

Strategy pattern

Where you used it

ballboy.model.Strategy.Strategy is the *strategy interface*.

ballboy.model.Strategy.HoldStillStrategy, ballboy.model.Strategy.GoRightStrategy and ballboy.model.Strategy.GoLeftStrategy are the *concrete strategy*.

ballboy.model.Enemy is the *context*.

ballboy.model.LevelImpl.tick() and ballboy.model.JSONInterpreter.createEnemy is the client.

What this pattern does for your code in terms of SOLID/GRASP principles

Maintains *Open/Closed Principle*, since new strategies can be introduced without breaking the code in the *context*.

What overall benefits this pattern provides (be specific to your code, not the pattern in general)

Strategy of Enemy can be switched during runtime, which is tick() in LevelImpl.

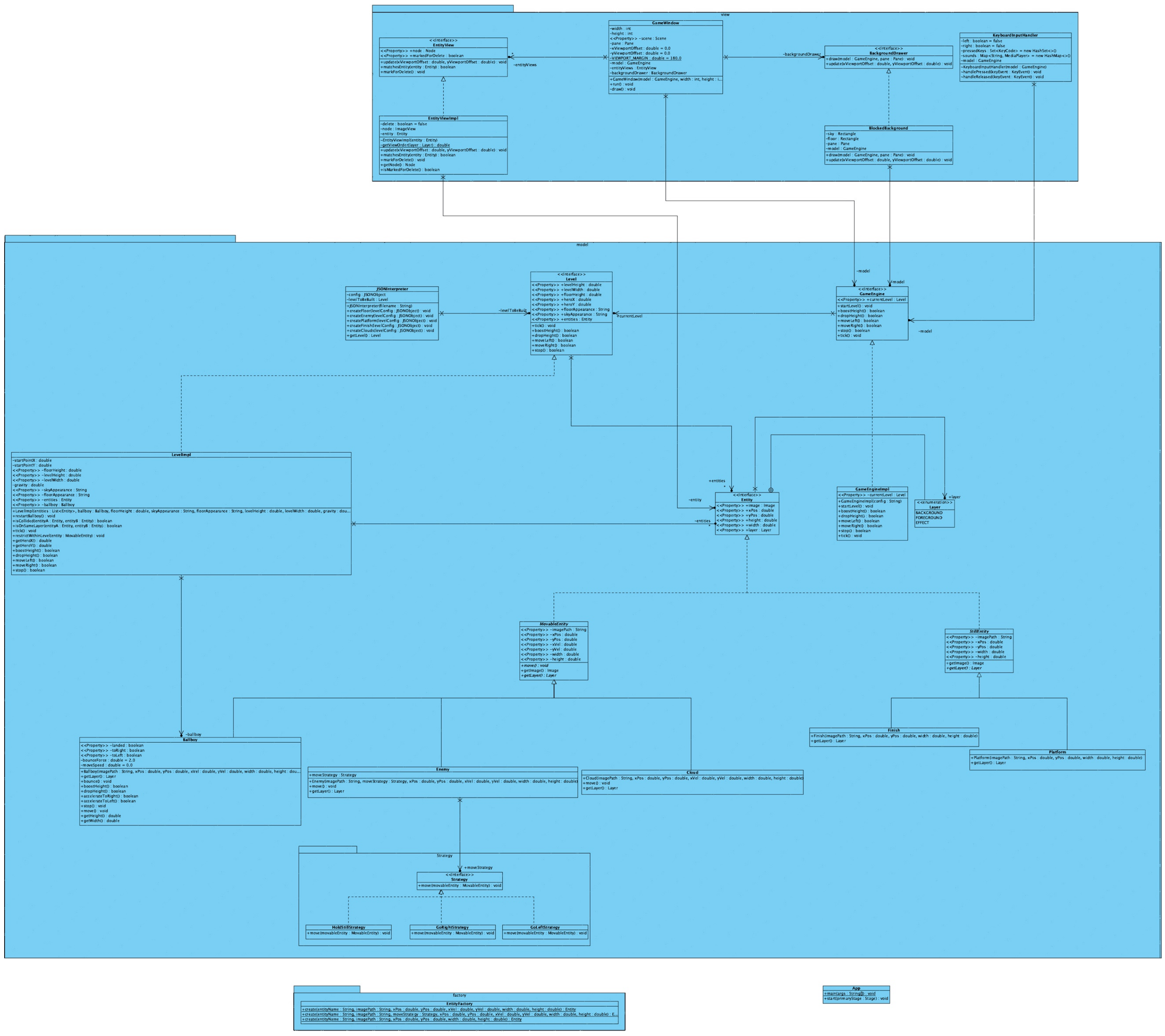
The details of the algorithm of the move strategy is isolated from Enemy.

What drawbacks this pattern causes

It might be overcomplicated since only three strategies has been introduced and they does not change.

Client must be aware of the difference between the MoveStrategy.

Assignment 2 updated UML class diagram


view
<<Interface>> EntityView
<<Property>> +node : Node
<<Property>> +markedForDelete : boolean
+update(xViewportOffset : double, yViewportOffset : double) : void
+matchesEntity(entity : Entity) : boolean
+markForDelete() : void
-entityViews
GameWindow
-width : int
-height : int
<<Property>> -scene : Scene
-pane : Pane
-xViewportOffset : double = 0.0
-yViewportOffset : double = 0.0
-VIEWPORT_MARGIN : double = 180.0
-model : GameEngine
-entityViews : EntityView
-backgroundDrawer : BackgroundDrawer
+GameWindow(model : GameEngine, width : int, height : i...
+run() : void
-draw() : void
-backgroundDrawer
<<Interface>> BackgroundDrawer
+draw(model : GameEngine, pane : Pane) : void
+update(xViewportOffset : double, yViewportOffset : double) : void
KeyboardInputHandler
-left : boolean = false
-right : boolean = false
-pressedKeys : Set<KeyCode> = new HashSet<>()
-sounds : Map<String, MediaPlayer> = new HashMap<>()
-model : GameEngine
~KeyboardInputHandler(model : GameEngine)
~handlePressed(keyEvent : KeyEvent) : void
~handleReleased(keyEvent : KeyEvent) : void
EntityViewImpl
-delete : boolean = false
-node : ImageView
-entity : Entity
~EntityViewImpl(entity : Entity)
-getViewOrder(layer : Layer) : double
+update(xViewportOffset : double, yViewportOffset : double) : void
+matchesEntity(entity : Entity) : boolean
+markForDelete() : void
+getNode() : Node
+isMarkedForDelete() : boolean
BlockedBackground
-sky : Rectangle
-floor : Rectangle
-pane : Pane
-model : GameEngine
+draw(model : GameEngine, pane : Pane) : void
+update(xViewportOffset : double, yViewportOffset : double) : void
model
JSONInterpreter
-config : JSONObject
-levelToBeBuilt : Level
+JSONInterpreter(filename : String)
+createFloor(levelConfig : JSONObject) : void
+createEnemy(levelConfig : JSONObject) : void
+createPlatform(levelConfig : JSONObject) : void
+createFinish(levelConfig : JSONObject) : void
+createClouds(levelConfig : JSONObject) : void
+getLevel() : Level
-levelToBeBuilt
<<Interface>> Level
<<Property>> +levelHeight : double
<<Property>> +levelWidth : double
<<Property>> +floorHeight : double
<<Property>> +heroX : double
<<Property>> +heroY : double
<<Property>> +floorAppearance : String
<<Property>> +skyAppearance : String
<<Property>> +entities : Entity
+tick() : void
+boostHeight() : boolean
+dropHeight() : boolean
+moveLeft() : boolean
+moveRight() : boolean
+stop() : boolean
+currentLevel
-model
-model
-model
<<Interface>> GameEngine
<<Property>> +currentLevel : Level
+startLevel() : void
+boostHeight() : boolean
+dropHeight() : boolean
+moveLeft() : boolean
+moveRight() : boolean
+stop() : boolean
+tick() : void
LevelImpl
-startPointX : double
-startPointY : double
<<Property>> -floorHeight : double
<<Property>> -levelHeight : double
<<Property>> -levelWidth : double
-gravity : double
<<Property>> -skyAppearance : String
<<Property>> -floorAppearance : String
<<Property>> -entities : Entity
<<Property>> -ballboy : Ballboy
+LevelImpl(entities : List<Entity>, ballboy : Ballboy, floorHeight : double, skyAppearance : String, floorAppearance : String, levelHeight : double, levelWidth : double, gravity : dou...
+restartBallboy() : void
+isCollided(entityA : Entity, entityB : Entity) : boolean
+isOnSameLayer(entityA : Entity, entityB : Entity) : boolean
+tick() : void
+restrictWithinLevel(entity : MovableEntity) : void
+getHeroX() : double
+getHeroY() : double
+boostHeight() : boolean
+dropHeight() : boolean
+moveLeft() : boolean
+moveRight() : boolean
+stop() : boolean
+entities
*
-entity
-entities
*
<<Interface>> Entity
<<Property>> +image : Image
<<Property>> +xPos : double
<<Property>> +yPos : double
<<Property>> +height : double
<<Property>> +width : double
<<Property>> +layer : Layer
GameEngineImpl
<<Property>> -currentLevel : Level
+GameEngineImpl(config : String)
+startLevel() : void
+boostHeight() : boolean
+dropHeight() : boolean
+moveLeft() : boolean
+moveRight() : boolean
+stop() : boolean
+tick() : void
+layer
<<enumeration>> Layer
BACKGROUND
FOREGROUND
EFFECT
MovableEntity
<<Property>> -imagePath : String
<<Property>> -xPos : double
<<Property>> -yPos : double
<<Property>> -xVel : double
<<Property>> -yVel : double
<<Property>> -width : double
<<Property>> -height : double
+move() : void
+getImage() : Image
+getLayer() : Layer
StillEntity
<<Property>> -imagePath : String
<<Property>> -xPos : double
<<Property>> -yPos : double
<<Property>> -width : double
<<Property>> -height : double
+getImage() : Image
+getLayer() : Layer
-ballboy
Ballboy
<<Property>> -landed : boolean
<<Property>> -toRight : boolean
<<Property>> -toLeft : boolean
-bounceForce : double = 2.0
-moveSpeed : double = 0.0
+Ballboy(imagePath : String, xPos : double, yPos : double, xVel : double, yVel : double, width : double, height : dou...
+getLayer() : Layer
+bounce() : void
+boostHeight() : boolean
+dropHeight() : boolean
+accelerateToRight() : boolean
+accelerateToLeft() : boolean
+stop() : void
+move() : void
+getHeight() : double
+getWidth() : double
Enemy
+moveStrategy : Strategy
+Enemy(imagePath : String, moveStrategy : Strategy, xPos : double, yPos : double, xVel : double, yVel : double, width : double, height : double)
+move() : void
+getLayer() : Layer
Cloud
+Cloud(imagePath : String, xPos : double, yPos : double, xVel : double, yVel : double, width : double, height : double)
+move() : void
+getLayer() : Layer
Finish
+Finish(imagePath : String, xPos : double, yPos : double, width : double, height : double)
+getLayer() : Layer
Platform
+Platform(imagePath : String, xPos : double, yPos : double, width : double, height : double)
+getLayer() : Layer
Strategy
+moveStrategy
<<Interface>> Strategy
+move(movableEntity : MovableEntity) : void
HoldStillStrategy
+move(movableEntity : MovableEntity) : void
GoRightStrategy
+move(movableEntity : MovableEntity) : void
GoLeftStrategy
+move(movableEntity : MovableEntity) : void
factory
EntityFactory
+create(entityName : String, imagePath : String, xPos : double, yPos : double, xVel : double, yVel : double, width : double, height : double) : Entity
+create(entityName : String, imagePath : String, moveStrategy : Strategy, xPos : double, yPos : double, xVel : double, yVel : double, width : double, height : double) : E...
+create(entityName : String, imagePath : String, xPos : double, yPos : double, width : double, height : double) : Entity
App
+main(args : String[]) : void
+start(primaryStage : Stage) : void