

# Project 7 Update - CSCI 4448 - Breadfest

## **1. Status Summary**

**Project Title:** The Quest for Breadfest

**Names:** Gustav Cedergrund & Joseph Allred

**Work Done:** The vast majority of internal game logic has been developed! Gustav Cedergrund built the dice classes and factory, dinosaur classes and factory, ingredient classes and factory, in addition to implementing fundamental game logic about how rooms are developed and connected. Joseph Allred experimented with the graphic representation of the game, and worked on developing FXML files to represent Room objects. He successfully completed a script that could process a room adapter and wipe and write to a dynamic FXML file with the details of the room, so that the current room would be displayed. However, when this functionality was applied to the Room objects in the cave, we encountered issues with JavaFX's application library being unable to update live changes in FXML files with preexisting stages. For these reasons we had to scrap this plan halfway through our project design, and instead make a sceneBuilder that dynamically changed buttons, backgrounds, etc appropriately.

**Changes & Issues Encountered:** The biggest issue we encountered were some complication with JavaFX's Application library, which had a complicated process of building a stage, scene, and root for each application and forcing that application to launch before we could write our room content from the adaptor to the FXML files. We considered scrapping the whole random generation of rooms and hard coding a predefined number of rooms in a variety of FXML files, but instead we redesigned our graphical display to build new scenes for each room instead of building new FXML files. This way onActions that were referenced by buttons could have live feedback into the scene displayed. It was unfortunate having to rework a majority of Joseph's side of the project, but we were able to salvage the majority of the functionality and the preexisting room design he had composed.

**Patterns:**

Adaptor Pattern: Our adaptor pattern is crucial for enforcing encapsulation and serving as our translator between the cave game object and our display / user interface. For example, two of our most critical function `getRoomExitDirections()` and `getObjectsAtAllLocations()` can be referenced in the adaptor to pass room state information such as which buttons should be displayed, where dinosaur objects exist in the room, where ingredients exist in the room, etc.

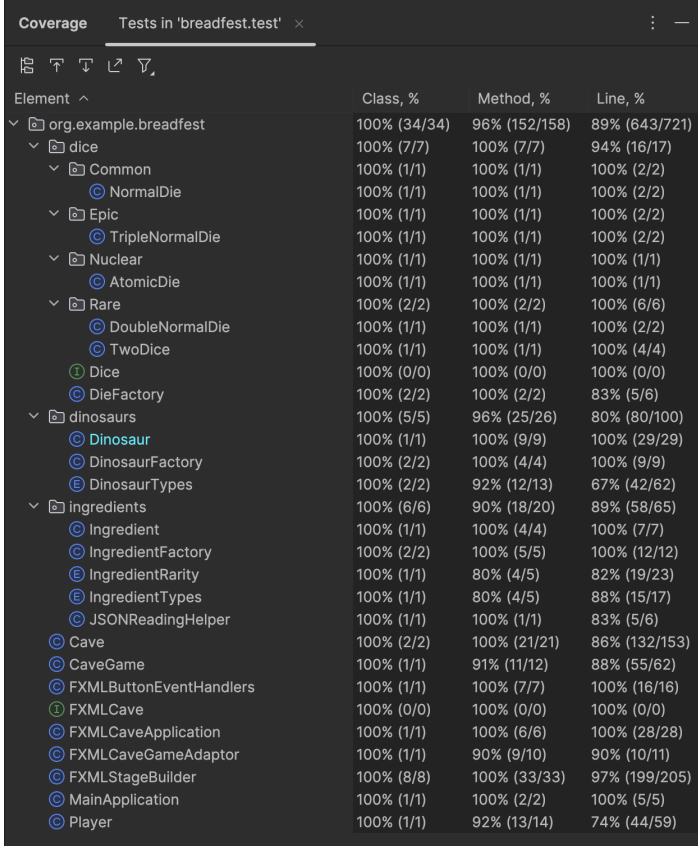
Builder Pattern: Our builder pattern is used for creating the scenes that are then displayed in the UI. This includes adding buttons to different places of a cave,

adding various images, and more. This makes code much more readable and lets us to stay true to DRY.

**Factory Pattern:** Our factory pattern is used to create random instances of ingredients, dinosaurs, and dice as they are populated into caves and into player inventory.

**Singleton Pattern:** Our singleton pattern is useful for maintaining the fact that we have one, and only one, player. We eagerly create this player upon creating our game, but from there we are able to upgrade players with the same functions without worrying about if we are operating on different objects.

## Test Coverage:

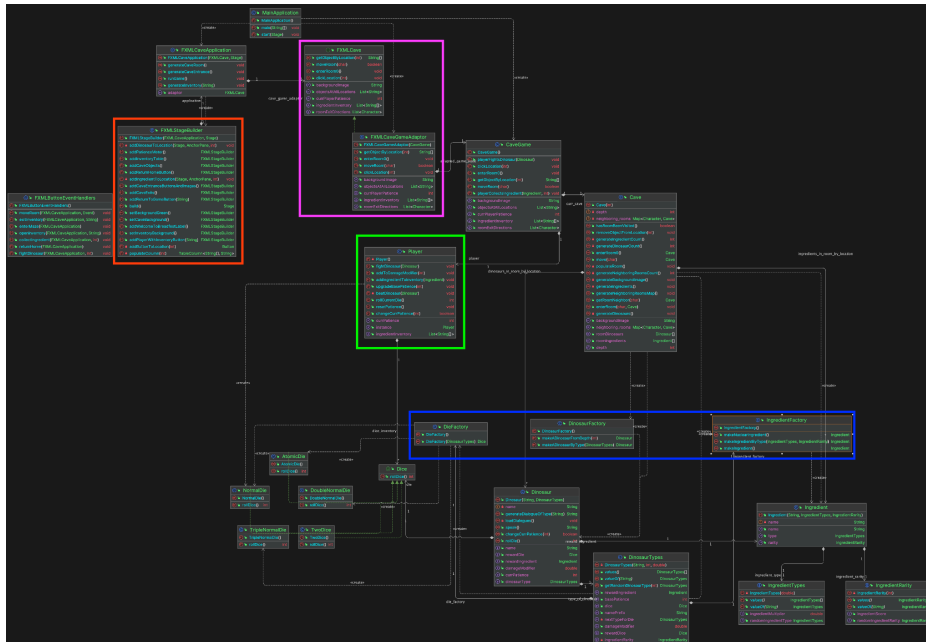


The screenshot shows the 'Coverage' tool in IntelliJ IDEA, displaying test results for 'Tests in 'breadfest.test''. The table lists various classes and methods, along with their coverage percentages and counts.

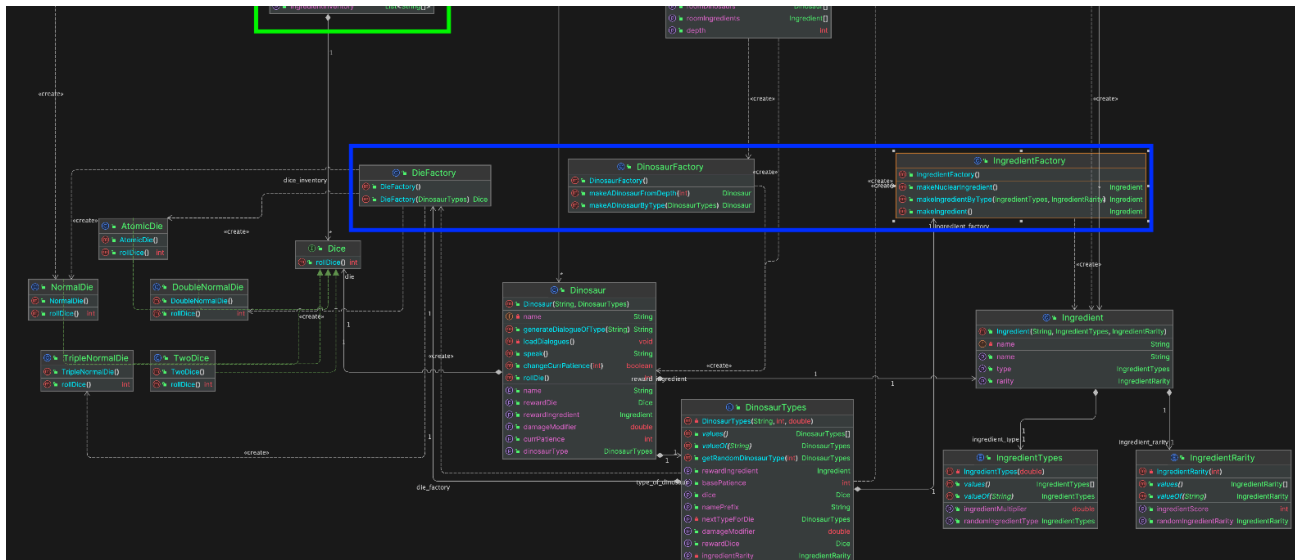
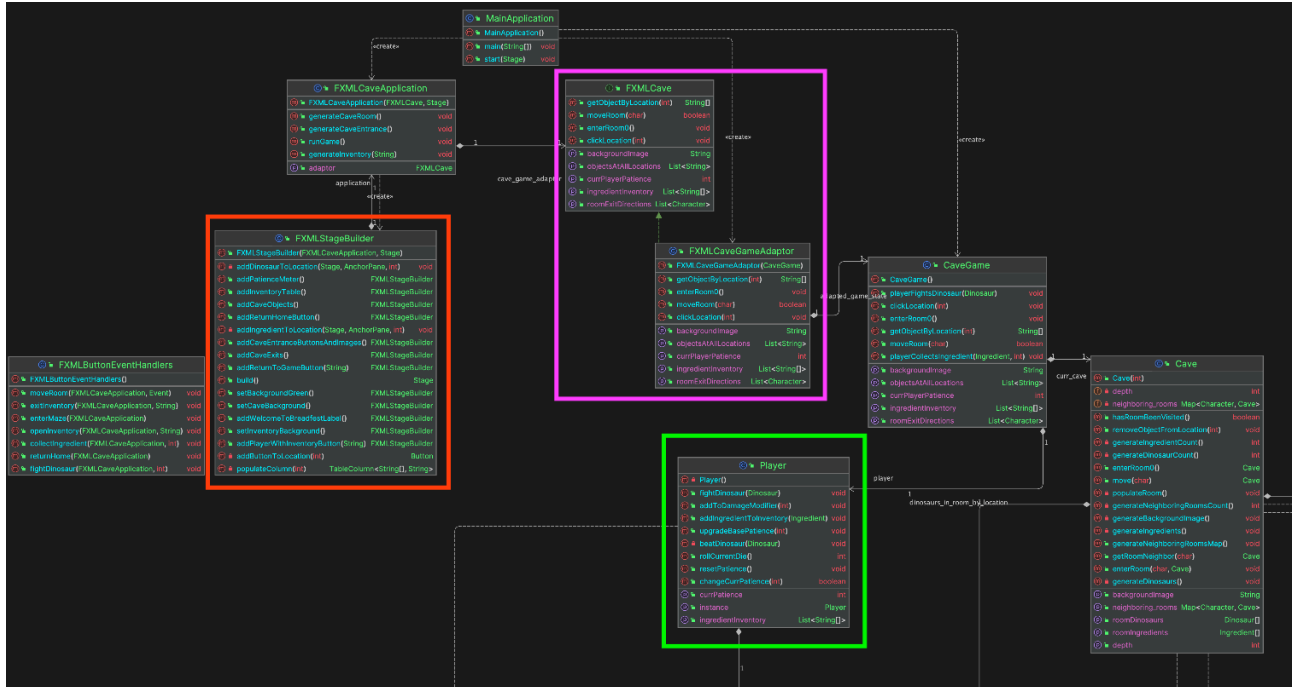
Element	Class, %	Method, %	Line, %
org.example.breadfest	100% (34/34)	96% (152/158)	89% (643/721)
dice	100% (7/7)	100% (7/7)	94% (16/17)
Common	100% (1/1)	100% (1/1)	100% (2/2)
NormalDie	100% (1/1)	100% (1/1)	100% (2/2)
TripleNormalDie	100% (1/1)	100% (1/1)	100% (2/2)
Nuclear	100% (1/1)	100% (1/1)	100% (1/1)
AtomicDie	100% (1/1)	100% (1/1)	100% (1/1)
Rare	100% (2/2)	100% (2/2)	100% (6/6)
DoubleNormalDie	100% (1/1)	100% (1/1)	100% (2/2)
TwoDice	100% (1/1)	100% (1/1)	100% (4/4)
Dice	100% (0/0)	100% (0/0)	100% (0/0)
DieFactory	100% (2/2)	100% (2/2)	83% (5/6)
dinosaurs	100% (5/5)	96% (25/26)	80% (80/100)
Dinosaur	100% (1/1)	100% (9/9)	100% (29/29)
DinosaurFactory	100% (2/2)	100% (4/4)	100% (9/9)
DinosaurTypes	100% (2/2)	92% (12/13)	67% (42/62)
ingredients	100% (6/6)	90% (18/20)	89% (58/65)
Ingredient	100% (1/1)	100% (4/4)	100% (7/7)
IngredientFactory	100% (2/2)	100% (5/5)	100% (12/12)
IngredientRarity	100% (1/1)	80% (4/5)	82% (19/23)
IngredientTypes	100% (1/1)	80% (4/5)	88% (15/17)
JSONReadingHelper	100% (1/1)	100% (1/1)	83% (5/6)
Cave	100% (2/2)	100% (21/21)	86% (132/153)
CaveGame	100% (1/1)	91% (11/12)	88% (55/62)
FXMLButtonEventHandlers	100% (1/1)	100% (7/7)	100% (16/16)
FXMLCave	100% (0/0)	100% (0/0)	100% (0/0)
FXMLCaveApplication	100% (1/1)	100% (6/6)	100% (28/28)
FXMLCaveGameAdaptor	100% (1/1)	90% (9/10)	90% (10/11)
FXMLStageBuilder	100% (8/8)	100% (33/33)	97% (199/205)
MainApplication	100% (1/1)	100% (2/2)	100% (5/5)
Player	100% (1/1)	92% (13/14)	74% (44/59)

## 2. Class Diagram

Below, we have included our UML Diagram in its entirety.



This Diagram is quite extensive, and entirely illegible, so we split this into two parts below. The patterns are highlighted in boxes: the factory patterns are in blue, the singleton pattern is in green, the builder pattern is in red, and the adapter pattern is in pink. Currently, we have created all of the classes for creating the UI, as well as the cave exploration portion of the game. To make the bread baking portion of the game, we will likely need to make a few more classes (to create a new UI display methods, a system of bread score calculation, etc.), but they will be incredibly straightforward as all of the base functionality is already implemented in our current classes (we already have a UI display builder and getScore methods for all ingredients).



### 3. BDD Scenarios

#### Feature 1: Beginning the Game Exploration

**Scenario:** The user enters the first room in the cave via the 'Enter Cave' button in the home room

**Given:** The user is in the home room with the cave entrance image and 'Enter Cave' button

**When:** The user clicks the 'Enter Cave' button

**Then:** The displayed scene changes and puts the user in the first room of the cave

## **Feature 2:** Navigating through the Cave

**Scenario:** The user moves between rooms via buttons displayed at room edges

**Given:** The user is in a room in the cave that has at least one directional button

**When:** The user clicks one of the direction buttons labeled 'Move North', 'Move South' etc

**Then:** The displayed scene changes and puts the user in the room adjacent to their previous room in the direction of the clicked button

## **Feature 3:** Collecting Ingredients in the Cave

**Scenario:** The user adds an ingredient in a room to their inventory

**Given:** The user is in a room in the cave that has at least one ingredient displayed next to the user

**When:** The user clicks on one of the respective ingredient images

**Then:** The displayed photo of the ingredient disappears

**Then:** The name, type, and rarity of the ingredient is added to the players inventory

## **Feature 4:** Gambling with Dinosaurs in the Cave

**Scenario:** The user enters the 'combat' scene with a dinosaur in their room

**Given:** The user is in a room in the cave that has at least one dinosaur displayed next to the user

**When:** The user clicks on one of the dinosaurs displayed next to them

**Then:** The user enters the 'combat' scene with the dinosaur where they can play dice

## **Feature 5:** Returning Home from the Cave

**Scenario:** The user returns to the home screen from any given room in the cave

**Given:** The user is in a room in the cave (any room but the home room with the cave entrance)

**When:** The user clicks the 'Return Home' button in the top right corner of the displayed scene

**Then:** The display changes from a cave room to the home room with the cave entrance

## **Feature 6:** Opening Inventory

**Scenario:** The user opens and views the contents of their inventory from a cave room or the home room

**Given:** The user does not have their inventory screen currently displayed

**When:** The user clicks the image of their avatar in the middle of the screen

**Then:** The current scene displayed will change to a scene displaying the contents of their inventory

**Then:** The inventory will be shown as a list of all the ingredients they've collected since beginning the game

## **4. Plan for Next Iteration**

→ **Modify this section last once we are ready to submit**

With the room objects, player objects, dinosaur objects, ingredient objects, and dice objects all built and functioning together with appropriate test coverage, our main focus moving forward is developing the fight functionality and display between the dinosaurs and player. Once we have a successful combat system implemented that can actively update the player's inventory with the items they win from the game, we plan to develop a very simple interface for baking the bread and receiving a score for it. This will be dependent on the value of the respective ingredients you have collected, which has already been constructed in our IngredientRarity enum. After these two implementations, most of our focus will be invested into better graphics for the home room, cave rooms, dinosaurs, buttons, etc. so the environment is more appealing.