**Xám**: Đã xong

Tuần 10 Chấm (18/7)

# Thầy: Làm nhỏ nhỏ thôi, quan trọng là áp dụng nhiều Pattern

Chuẩn bị 1 kịch bản game nhỏ, có thể trình bày = Slide. Trình bày rằng mình dùng Design Pattern đó vào việc gì.

*Nhặt ra 1 câu chuyện nhưng lồng 3-4 pattern vào nhau*

# Demo: 11/7

**Tuần 10 chấm Assignment: 18/7**

* ***Thứ 4, 6 tuần này chuyển sang thứ 2, 4 tuần 10***
* ***Nộp bài trước tuần 10 (trước thứ 2)***
* ***Chuẩn bị Slide đầy đủ, Mỗi nhóm 15p***
* ***Nhóm mình lên đầu***

# Design Pattern

## Command Pattern

Dùng để di chuyển main character: dùng các phím arrow

## State Design Pattern

Dùng cho game: (nhóm Đạt-Kiên-Hải-Thái Present)

+ VD: Trồng cây:

\* 1 ô đất sẽ có nhiều State (Chưa trồng, đã trồng, đã chín), và Player sẽ có nhiều hành động

\* Trồng: nếu ô đất đang có state là chưa trồng thì sẽ cho trồng, còn nếu đã trồng thì ko trồng tiếp dc

\* Tưới nước: nếu ô đất chưa trồng thì ko tưới dc, nếu đã trồng thì tưới dc, nếu đã chín thì ko tưới dc

\* Hái quả: ko làm dc nếu chưa trồng hoặc đã trồng nhưng chưa chín

+ Hữu dụng hơn khi có rất nhiều State, nhưng cx có thể dùng dc khi ít State: dùng để đóng gói Business của Object, khiến những thằng bên ngoài ko cần quan tâm đến hành động nội bộ của nó với từng state

Note:

* Tạo các Sprite khác nhau cho mỗi State của ô đất (nhờ Huyền) -> khi ô đất đổi state thì cx đổi sprite

## State Design Pattern

Dùng để điều khiển Transition Animation của MainCharacter: khi đi lên thì animation hướng lên trên, khi đi sang thì animation đi ngang…

## Decorator Pattern

* Ứng dụng vào Game Nông trại:
  + Super class: cây trồng
  + Décor: phân bón
    - Đạm Decor
    - Kali Decor
    - Lân Décor
  + => Mỗi Décor đều cầm thêm 1 Con trỏ của cây trồng
* => lúc đầu: new Cây
* Về sau: new Kali => đính Cây vào
* Khi cho Cây lớn lên -> phải dùng con trỏ của Kali vào
* Tóm lại:
  + Có 1 class **Chung** lớn, chung để các class khác extend
  + Những class nhân (Cây, rau) phải extend class lớn
  + Có 1 class Décor lớn (abstract) để các class Décor extend, và class này cx extend class **Chung** lớn
    - Có 1 attribute là 1 biến thuộc kiểu class **Chung**
  + Các class Décor sẽ extends class Décor lớn
* Khi tạo 1 object:
  + Tạo 1 object nhân
  + Mỗi khi muốn thêm 1 Décor:
    - Tạo 1 object Décor, với attribute thuộc kiểu class **Chung** là reference đến object nhân
  + Mỗi khi muốn thêm 1 Décor:
    - Tạo object décor nữa r reference đến object décor kia
* Dùng để làm món ăn (thêm topping)

## Object Pool Pattern

* Trời mưa: điều khiển các giọt mưa
* *Kết hợp Flyweight*

## Abstract Factory Pattern

Sản sinh ra các object giống (cùng loại) nhau:

* Các loại cây trồng khác nhau
* Các con vật khác nhau

## Observer Pattern

* Dùng làm Event - bắt những sự kiện bấm nút đặc biệt của ng chơi (trồng cây, tưới nước,...)

# Algorithms

*(cần có* ***Big-O****)*

## Shortest Path

Dùng để di chuyển theo kiểu bấm chuột vào màn hình

(Tham khảo: Xem lại bài tập Graph – Shortest Path)

# Data Structures

## Graph (Undirected Weighted):

Làm cùng với bài Shortest Path

## Linked List (Sorted Linked List):

Làm cùng với bài Shortest Path

## Queue:

Để lưu list các hạt giống mua dc -> mỗi khi lấy ra trồng thì lấy cái cũ nhất (xem code Dr.T)

# File I/O

## Lưu dữ liệu vào File:

* Số tiền của Nhân vật

# Others

## Game Design:

* Bản đồ bao gồm các ô đất và các con đường đc sắp xếp theo hình bàn cờ
  + Các con đường và ô đất chưa mua sẽ chỉ là background (ô đất thì có collider)
  + Sau khi ng dùng chọn mua ô đất mới trong cửa hàng (có thể cho bấm chuột để chọn vị trí cho ô đất đó) -> instantiate ra 1 ô đất (farm plot) mới ở vị trí đó (vị trí xác định cho trc)
  + Các con đường xung quanh các ô đất -> khi bấm vào 1 điểm trên con đường -> xác định ô đường của điểm nhấn -> di chuyển nhân vật chính đến ô đó = Shortest Path Algorithm

## Background & Map Creation:

Sử dụng **TileMap**

* Guide:
  + <https://viblo.asia/p/huong-dan-su-dung-tilemap-tao-mot-map-trong-unity-bWrZnwavlxw>
  + <https://learn.unity.com/tutorial/introduction-to-tilemaps>