Design Patterns

Final Assignment

Date: 08/02/2024

Students: Jonas Ramanauskas 4681789, Caner Celik, 4594738

Contents

Introduction

The final project aims to create a Java application using at least 4 different design patterns. The chosen patterns include:

- Decorator
- State
- Observer
- Factory
- Singleton

The team decided to build a small Command Line Interface application as a demonstration of these patterns. The application isn't meant to be practical; it simply shows the team's ability to apply these design patterns in Java.

Description

In the Moscow method, the emphasis is on defining features based on their priority: Must-haves (M), Should-haves (S), Could-haves (C), and Won't-haves (W).

Must-haves (M):

- Players oversee agricultural enterprise activities.

- Players act as farm managers responsible for overseeing farm activities, recruiting workers, and harvesting crops.

- Workers have meters for fatigue, hunger, and total hours worked.

- Workers retire if fatigue, hunger, or total hours worked exceed a certain threshold.

- Players can hire field workers or harvesters.

- Completed harvests must be transported to the storage area for processing.

Should-haves (S):

- Workers can be assigned additional tasks beyond their primary responsibilities.

Could-haves (C):

- Implementing more detailed worker behaviors and interactions.

- Introducing seasonal variations affecting crop growth and worker efficiency.

- Adding upgrades or improvements to the farm infrastructure.

Won't-haves (W):

- Complex economic models governing market prices and demands.

- Detailed weather systems impacting crop growth.

By categorizing features using the Moscow method, we will prioritize development efforts and ensure that the essential elements of our application implements first while leaving room for potential enhancements in the future.

Design Patterns to be realized

Decorator Pattern

- This pattern could be used to dynamically add additional responsibilities to workers beyond their primary ones. For example, you might have a base worker class and then decorators for specific tasks or abilities that can be added on top of that base behavior.

State Pattern

- The State pattern could be applied to manage the various states of workers, such as working, resting, or retired. It could also be used to handle the different states of crops, such as planted, growing, ready for harvest, etc.

Observer Pattern

- The Observer pattern could be utilized to monitor the state of workers' metrics (fatigue, hunger, total hours worked) and trigger actions when these metrics exceed certain thresholds. For instance, the farm manager could observe the state of each worker and react accordingly if any worker needs attention.

Factory Pattern

- The Factory pattern could be employed to create different types of workers (field workers, harvesters) or crops in a centralized way. It abstracts the process of object creation and allows the client code (in this case, the farm manager) to interact with different types of objects through a common interface.

Singleton Pattern

- The Singleton pattern might be used to ensure that there's only one instance of critical classes within the simulation. For instance, you might have a single instance of a FarmManager class that oversees the entire simulation to ensure consistency and prevent multiple managers from conflicting with each other.

Each of these design patterns serves a specific purpose in software design and can help improve the structure, flexibility, and maintainability of the applications codebase by encapsulating different aspects of the system.

Version Control

Git, a free and open-source distributed version control system or protocol, will be utilized. It will help in tracking changes to files over time and afterwards view those changes as well as specific versions of those files.

GitHub user interface application will be used as a client program that's easy to use. It can be used to see and edit the website's code.