Tetris in java

This is a java app about a project for CS242

Abstract

Project Purpose

Learning OOP programming in java

Project Motivation

Tetris is a famous game in a long time, I would like to realize it in java.

Technical Specification

- Platform: Java Eclipse
- Programming Languages: Java
- Stylistic Conventions: Java styling conventions

Functional Specification

Features

- Follow all the rules of traditional Tetris, players can rotate pieces, when a row is fulfilled, it will be cancelled and players will get score
- Different piece have different colors, same color will gain more score
- Different levels (piece drop speed & frequency) according to current scores
- Functions include pause, Restart, jump to next level and manipulation of highest score record.
- Custom pieces

Brief Timeline

- Week 1: Construct the basic web page and components with basic UI. Users should be able to put their information.
- Week 2: Implement hashtags and search functions for hashtags. System only store 1000 most recent posts.
- Week 3: Improve UI, use apis to allow users get covid19 information nearby.

Rubrics

Week 1

Category	Total Score Allocated	Detailed Rubrics
Create Pieces	4	0: No pieces created 2: Some pieces can not be created 4: create at least 4 kinds of pieces with blocks in different shapes
Function Rotate up	4	0: No pieces can be rotated 2: some pieces cannot rotate correctly 4: Rotate the pieces clockwise with 90 degree
Function Drop Down	3	0: Didn't implement anything 3: Pieces will move down 1 unit
Function Left	3	0: Didn't implement anything 3: Pieces will move left 1 unit

Function Right		0: Didn't implement anything 3: Pieces will move Right 1 unit
Testing - Tetris Pieces, Game Logic	6	0: Didn't implement tests 3: when JUnit tests for all Tetries behaviors implemented have coverage above 50% 6: when JUnit tests for all Tetries behaviors implemented have coverage above 80% 6: coverage above 90%
Test Data Structures	5	0: Didn't implement tests 2: JUnit tests for all Tetris piece behaviors implemented have coverage above 50% 5: JUnit tests for all Tetris piece behaviors implemented have coverage above 80%

Week 2

Category	Total Score Allocated	Detailed Rubrics
Keyboard interaction	6	0: Didn't implement anything 3: Some keys not working interaction 6: Up key to rotate and left/right/down key to move
Static User Interface	8	0: Didn't implement anything 4: The UI consists of a neat layout that accurately represents a Tetris playboard 6: Implement GUI instead of terminal UI 8: UI consist of function ui include restart, pause and score.

Refactoring/ Completing missing functions	4	0: Didn't implement anything 1 point - The code still contains a lot of duplicate/redundant code with room for a lot of improvements. OR The code still lacks functionality from last week 2 points - There is not much duplicate/redundant code and the student has put effort to refactor code from last week. AND The code also contains all requirements from the previous week. However, there is still room for improvement. 4 points - Every piece of code is functional and the overall design of the project is so well designed that it cannot be refactored/improved any further.
Testing	6	Unit tests are written for all new code & expanded last week's tests if necessary 0: coverage below 30% 3: coverage above 50% 5: coverage above 75% 6: coverage above 90%
Manual Test Plan	4	0: no MTP 1 pts if the test plans include only environment setup OR scenario descriptions 3: more than 4 pages that contain most of the content 4: Well-composed test plans

Week 3

Category	Total Score Allocated	Detailed Rubrics
Function Game Loop	8	0: Didn't implement anything 5: The program works well, pieces can be rotated and cancelled when row is fulfilled 7: Game level could change manually 8: Game will store the highest record, players can clean the record
Refactoring/ Completing missing functions	5	4 points - There is not much duplicate/redundant code and the student has put effort to refactor code from last week. AND The code also contains all requirements from the previous week. However, there is still room for improvement. 5 points - Every piece of code is functional and the overall design of the project is so well designed that it cannot be refactored/improved any further.
Function pause, level up, restart	6	0: Didn't implement anything 6: 2 pts for each function (level up means game will automatically change level based on current scores)
Testing - Unit Tests for New Code	6	0: Didn't implement tests 6: can render pages correctly

Testing - Manual Test Plan	5	0: no manual test plan 2: basic plan in words 4: 8 pages of plan contain most information 5: 10 pages of well-defined plan