AP Project

Deadline-FINAL SUBMISSION

TEAM:

NOEL 2022338
MUTHURAJ 2022307

GROUP NO:22

COMPONENTS:

- HOW TO PLAY THE GAME
- BONUS COMPONENT
- DESIGN PATTERNS
- JUNIT FEATURES
- UML DIAGRAMS

HOW TO PLAY THE GAME:

To play the game follow these steps:

To execute the game:

```
mvn clean javafx:run
```

1. Go to the Hello Application file and run the current file
2. You will find the start screen press the spacebar to start the game.
3. Now to create a stick hold on to the spacebar.
4. To flip the character downwards press the down-arrow button
5. You will encounter barrels randomly, colliding with the barrels will end the game.
6. To not collide with the barrels you have two options either you can flip the character or jump over the character.
7. To jump over the character press the spacebar button.
8. If the character dies you can either revive or choose to exit.However,note, to revive the character you should have earned a minimum of 10 bananas.
9. The player's score and Best score will be present in the ending screen once the player dies.
10. In the ending scene you will again have the option to go to home and play the game all over again.
11. The user's gameplay scores will be present in the end screen, along with the user's score and the highest score hence saving the gameplay details.

BONUS COMPONENT:

The following bonus component have been implemented:

1. Jump features: the character has the ability to jump over the barrels by pressing spacebar.
2. We have implemented barrels which act as barricade, to escape from barrel attacks the player can either choose to jump or flip upside down.
3. We have set a timer for the backgrounds to make sure the backgrounds change after a fixed interval of time(5 seconds).The backgrounds keep changing in a manner relating to the various durations of a day(from sunrise to sunset).
4. The bananas are in constant floating motion to create an overall effect.
5. When the player dies the game has a pop up option which asks whether the player wishes to revive or not. However,note,to revive the character you should have earned a minimum of 10 bananas.

DESIGN PATTERNS:

The following design patterns have been used in our applications:

1.Singleton pattern for score tracker (Class: Score_Tracking, Line 40):

We have created a single instance for score tracker.

2.Decorator pattern for Reading and Writing File:

We have utilized buffered reader attribute in file_reader() function.

JUNIT FEATURES:

We have implemented two instances of Junit classes.

Instance 1:

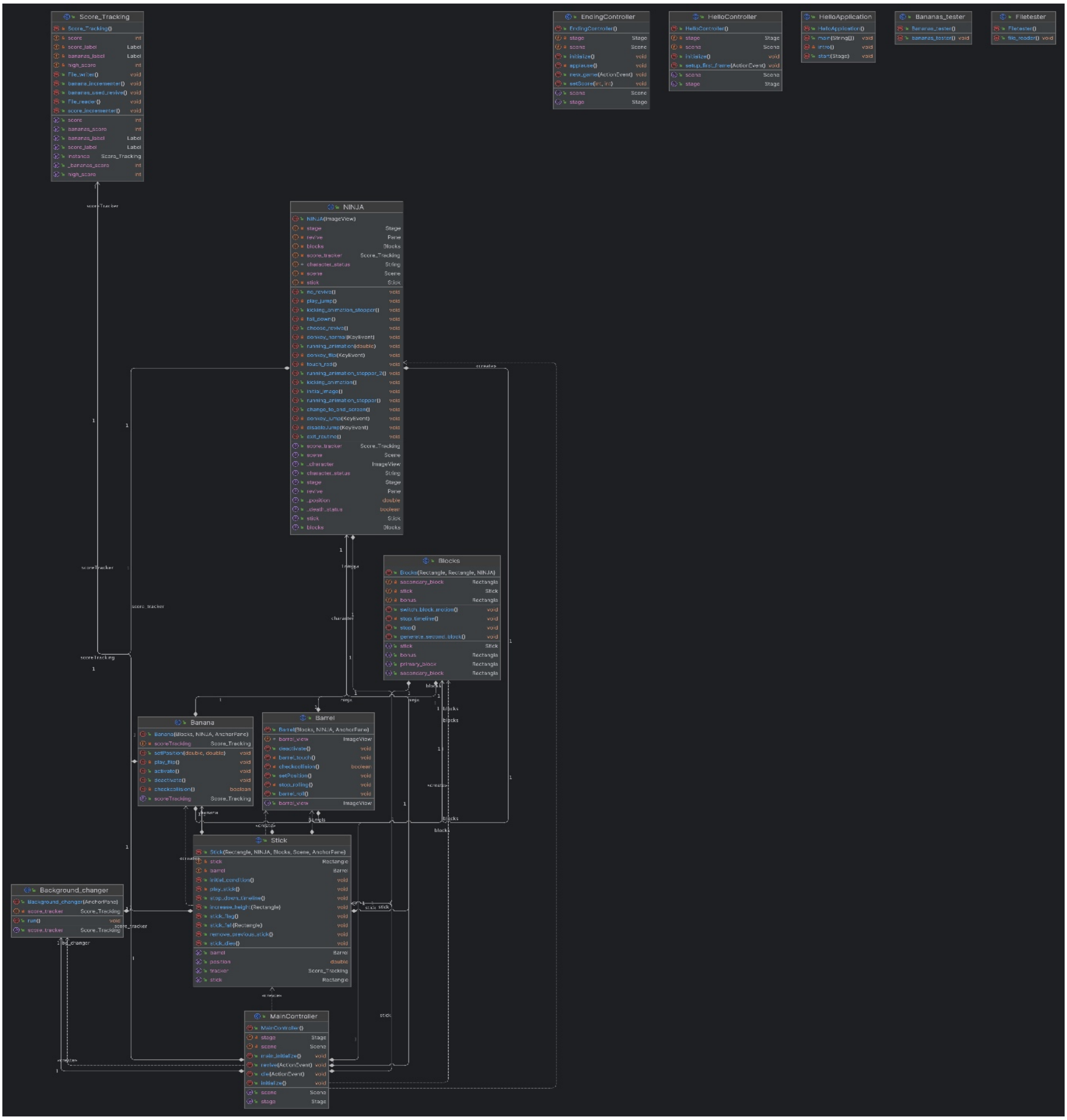
- To make sure the smooth transitioning of the score tracker happens we have added a caution junit tester for the file function, we always make sure the required values for the user score and the player's best score is present in the file. We have used the assert Equals function and compared it with the file-size,the assertion throws false if it doesn't detect the required data otherwise the testcase is passed. This make sure that the scores are getting properly tracked and stored.

Instance 2:

- To make sure the reviving functionality works properly without any hassle, we have again added another caution junit tester.
- The purpose of this function deals with condition that reviving happens only when the user has accumulated more than 10 bananas throughout his gameplay.
- If the user has not earned 10 or more bananas then the code our code contradicts this by checking with assert Equals functionality. The assert Equals functionality makes sure the user has a minimum of 10 bananas for the reviving functionality to work.

- If the user doesn’t then an assertion is thrown claiming false otherwise the testcases end up positive.

UML DIAGRAMS:


Score_Tracking
Score_Tracking()
score int
score_label Label
bananas_label Label
high_score int
File_writer() void
banana_incrementer() void
bananas_used_revive() void
File_reader() void
score_incrementer() void
score int
bananas_score int
bananas_label Label
score_label Label
instance Score_Tracking
_bananas_score int
high_score int
EndingController
EndingController()
stage Stage
scene Scene
initialize() void
applause() void
new_game(ActionEvent) void
setScore(int, int) void
scene Scene
stage Stage
HelloController
HelloController()
stage Stage
scene Scene
initialize() void
setup_first_frame(ActionEvent) void
scene Scene
stage Stage
HelloApplication
HelloApplication()
main(String[]) void
intro() void
start(Stage) void
Bananas_tester
Bananas_tester()
bananas_tester() void
Filetester
Filetester()
file_reader() void
NINJA
NINJA(ImageView)
stage Stage
revive Pane
blocks Blocks
score_tracker Score_Tracking
character_status String
scene Scene
stick Stick
no_revive() void
play_jump() void
kicking_animation_stopper() void
fall_down() void
choose_revive() void
donkey_normal(KeyEvent) void
running_animation(double) void
donkey_flip(KeyEvent) void
touch_red() void
running_animation_stopper_2() void
kicking_animation() void
initial_image() void
running_animation_stopper() void
change_to_end_screen() void
donkey_jump(KeyEvent) void
disableJump(KeyEvent) void
exit_routine() void
score_tracker Score_Tracking
scene Scene
_character ImageView
character_status String
stage Stage
revive Pane
_position double
_death_status boolean
stick Stick
blocks Blocks
Blocks
Blocks(Rectangle, Rectangle, NINJA)
secondary_block Rectangle
stick Stick
bonus Rectangle
switch_block_motion() void
stop_timeline() void
stop() void
generate_second_block() void
stick Stick
bonus Rectangle
primary_block Rectangle
secondary_block Rectangle
Banana
Banana(Blocks, NINJA, AnchorPane)
scoreTracking Score_Tracking
setPosition(double, double) void
play_flip() void
activate() void
deactivate() void
checkcollision() boolean
scoreTracking Score_Tracking
Barrel
Barrel(Blocks, NINJA, AnchorPane)
barrel_view ImageView
deactivate() void
barrel_touch() void
checkcollision() boolean
setPosition() void
stop_rolling() void
barrel_roll() void
barrel_view ImageView
Stick
Stick(Rectangle, NINJA, Blocks, Scene, AnchorPane)
stick Rectangle
barrel Barrel
initial_condition() void
play_stick() void
stop_down_timeline() void
increase_height(Rectangle) void
stick_flag() void
stick_fall(Rectangle) void
remove_previous_stick() void
stick_dies() void
barrel Barrel
position double
tracker Score_Tracking
stick Rectangle
Background_changer
Background_changer(AnchorPane)
score_tracker Score_Tracking
run() void
score_tracker Score_Tracking
MainController
MainController()
stage Stage
scene Scene
main_initialize() void
revive(ActionEvent) void
die(ActionEvent) void
initialize() void
scene Scene
stage Stage
scoreTracker
score_tracker
scoreTracking
ninja
character
blocks
banana
barrels
stick
bg_changer
«create»
1