# Code Review

By Isaac & Josiah

For our project we chose Chrome Dinosaur:
https://github.com/ImKennyYip/chrome-dinosaur-java

Below is the class documentation we wrote up concerning the classes for the program.

# ChromeDinosaur.class

## Class variables

[**int**] Boardwidth, Boardheight.--- Contains the preferred dimensions of the board.
[**Image**] dinosaurImage, dinosaurDeadImage, cactus 1, cactus 2, cactus 3 — holds the image of the variables
[**class**] **Block** (Holds the coordinates, image, and size of an image)
[**ints**] dinosaurWidth, height, x-position, y-position — holds the values for a dinosaur object
[int] cactusWideth, height, x-position, y-position — holds the values for a created cactus object

### Make an arrayList of Cactuses

- This initializes an empty list of Cactus Block objects to be filled when placeCactus() is called.

### Create the physics variables

- velocityX - speed of Cactus moving towards you.
- velocityY - The speed of the dinosaur falling toward the ground
- Gravity - accelerates the velocityY toward the ground

### Timers

- Gameloop - creates the game at 60fps
- placeCactusTimer - calls placeCactus() every 1.5 seconds creating a chance for a cactus obstacle to be created.

# Constructor

- Sets the background size, background colour
- Makes the dinosaur focusable for Java.awt
- Makes the object a keyListener
- Initialises all the class variables images to their respective images.
- Sets Gamespeed to 60fps.
- Starts the game.
- Creates a delay, and then every 1.5 seconds calls placeCactus
- Starts the placeCactus timer.

# Functions

## Place Cactus

**Parameters**: None
**Returns**: void

- If the game is over, return nothing.
- Randomly creates 1 of 3 cacti.
    - 10% chance the cactus is very big
    - 20% chance it is medium
    - 20% change it is small
    - 50% if it is nothing.
- If there's more than 10 cacti in the **queue**, we remove them.

## Paint Component

**Parameters**: Graphic g
**Returns**: void

Calls the super method for Jpanel, while also calling the class method draw.

## Draw

**Parameters**: Graphic g
**Returns**: void

- Uses the Graphics attribute method drawImage() to draw the dinosaur.
- Draws all the cacti.
- If the Game is over, displays the game over credits: otherwise it draws the score.

Returns null

## Move

**Parameters:** None
**Returns:** Void

- Changes the velocity of the Dinosaur
- Checks to see if the Dinosaur is on the ground, if so sets the y-velocity to 0.
- For every cactus, changes the x-position while doing so checks to see if there is a collision; if there is a collision, sets gameOver to True and changes the Dinosaur image to dinosaurDeadImg. Increments the score.

## Collision

**Parameters: Two block objects, a & b.**
**Return:** The boolean value of whether any corners of either object are touching.

## actionPerformed

**Parameters:** ActionEvent e
**Return:** Void

- Calls *move()*.
- Calls repaint() [This is a java.awt component]
- If the game is over, stops all timers (PlaceCactusTimer, gameLoop)

## keyPressed

**Parameters: KeyEvent e** (which key was pressed)
**Return:** void

- If the key pressed was SPACE
  - Check if Dinosaur is on the ground if it changes the dinosaur's y velocity to -17.
  - Changes dinosaurImg to dinosaurJumpImg.
- If the game is over
  - Reset all the class attributes in order to restart the game.

## keyTyped & KeyReleased - parent methods that are set to do nothing in this code.

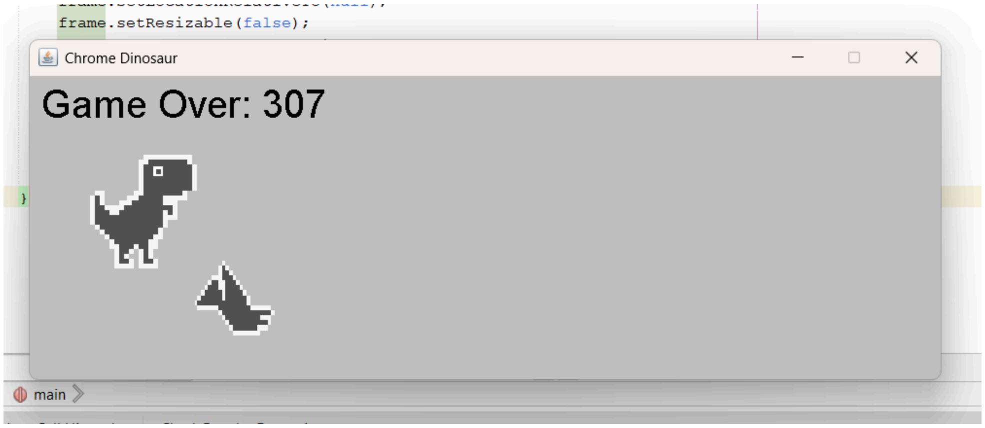
*dinosaurJu dinosaurJumpIm

# App.class

Creates a new Jframe.
Creates a new ChromeDinosaur Object.
Adds the ChromeDinosaur Object to our Frame.
Pack the frame (calculates the minimum space required to fit all frames in App frame)
Center the Frame upon the Chrome Dinosaur.
Make the frame visible.

## Our Changes

- **Created Birds as additional obstacles at various heights**. Our largest change to the game was creating bird obstacles. We created an array to be initialized with blocks of the bird image and produce these birds at random intervals and varying heights to increase difficulty.
- **Increased operation spread by storing fewer obstacles.** The initial program processed and graphed images that would not be on the screen. These images would be created even if they did not impact the user, so we changed the code so it would not display an excessive amount of obstacles.
- **Created track animation for dinosaur to appear to run along ground to improve graphics.** We created two rotating images of a track that rotate on the screen to create an illusion of the dinosaur running.
- **Changed scoring**. The original game kept score in a very ambiguous way, so we changed the score keeping function to track the time spent alive and display this score. This allows the user to identify what the actual value is rather than it being an imaginary number.
- **Display all-time high score**. We created an all-time high-score that appears during the game. This score turns green indicating you have beaten the all-time highest score when you surpass it and saves it for the attempt. Prior to beating the all-time high-score the high-score is displayed in red.
- **Debugged Code.** The initial game would sometimes have issues where it concluded the dinosaur collided with an obstacle and ended the game when clearly it did not (see image below). We rewrote the collision logic to prevent this from happening.



```
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
}
```

```
Game Over: 307
```