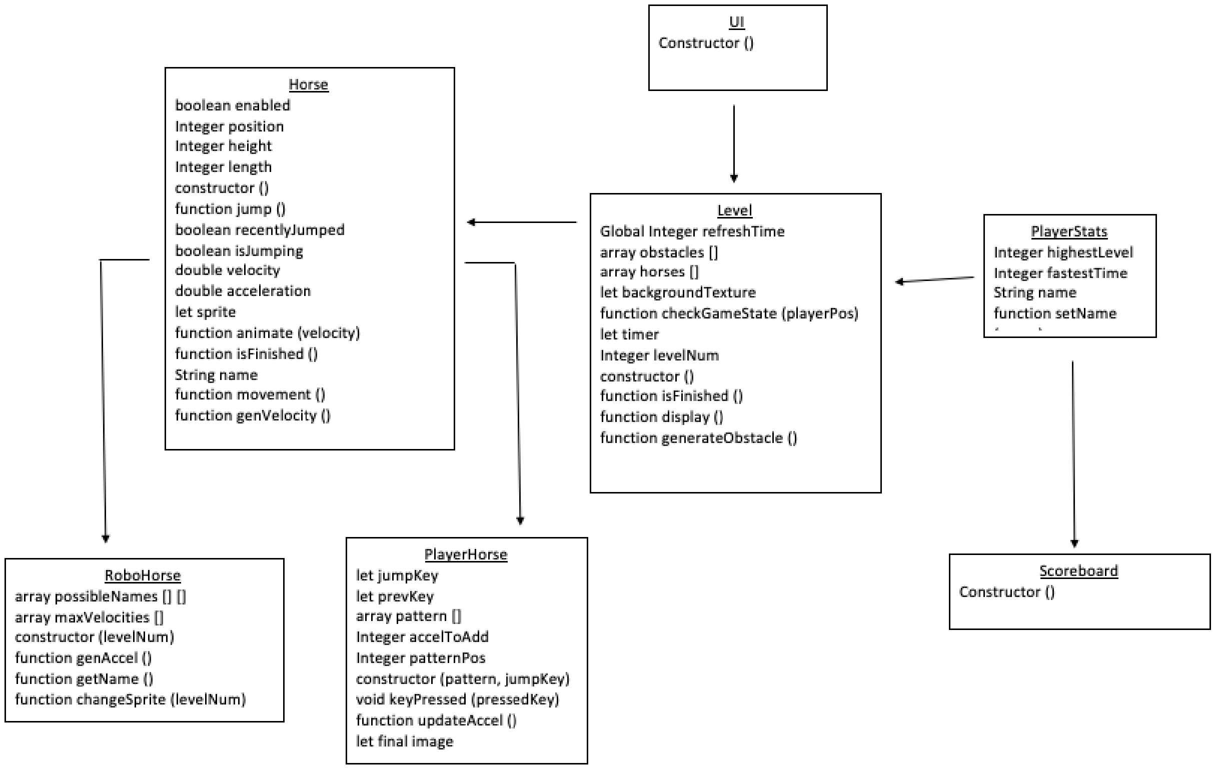
}



```
Class UI{
        constructor()//called when page loads
                Shows title screen image, starts first level automatically(calls level constructor)
                call checkGameState(level)
                        1 -> construct level 1
                        2 -> construct next level

}
class PlayerStats {
        Integer highestLevel  // keeps track of user's highest level reached
        Integer fastestTime // keeps track of user's fastest time
        String name  // keeps track of the player's inputted name
        function setName (String name) {
                User sets name, returns nothing
        }
}
```

```
class Scoreboard {
        //draws data from PlayerStats and displays information on top corner via text box, font
comic sans
        constructor() {
                Load and display highestLevel, fastestTime and name.
        }
}

class Level {
        Global integer refreshTime // the fps of the computer.
        array obstacles = [......] // array of booleans, bool is true when there is an obstacle there,
        array length 20000 // 1 meter corresponds to 100 indices

        array horsePerLanes // length 6, can be a playerHorse, roboHorse, or nothing in a lane
        (horse in each lane, except for final level)

        let backgroundTexture = image // each level can have a different background if desired

        function checkGameState (playerPos) { //run this every update frame

                Cross checks playerPos with obstacles when not jumping; returns an integer to
        indicate whether a collision (1), or finish(2), or neither has occurred(0).


        let timer // accurate to 0.01 seconds, keeps track of time from start of game

        int levelNum // number of the level; determines what the obstacles array and background
        is loaded as

        constructor {
                Load UI (scoreboard, backgroundTexture)
                based on levelNum, adds 1-5 roboHorses to lanes 1-5
                        at levelNum = 4, horse is just 1
                                level 1: 5 horses
                                level 2: 5 horses
                                level 3: 5 horses
                                level 4: 1 horse
                Make playerhorse at lane 6(index 5) // always at lane 6
                generateObstacle()
                Disable movement
                Cover race track with level starting image(depends on level)
                Wait 1sec
                Remove starting image
                Wait 1sec
```

```
            Start race(enable movement)
            Check isFinished every frame

        }
        function isFinished(){ //returns a boolean of whether the horse has reached the endline
            Return if pos equals length of obstacle array (false if not)
        }
        function display{//run every frame, handles graphic scrolling
            Get obstacles from index player position - 200 to player position + 200
            Show horse at center of screen
            Display obstacles at center of screen - 200 to + 200, at indices where obstacle =
true

        }
        Function generateObstacle () // evenly spaced obstacles
        {
        Integer div : Switch based on level number, (level : #num opticals) 1:4, 2:6, 3: 8, 4:12
            Make new array length 20000
            Loop 20000 times
                If current Math.floor(loop % (20000/div)) = 0
                    Set current index of array to true
                Else set to false

        }

}

class Horse {
        Boolean enabled //ALL functions (except constructor)only work if enabled is true
        Integer position // horse's position as represented in the array; a number between 0 and
        the length of the obstacles array.
        Integer length = 197 //width of single frame of sprite
        Integer height = 158 //height of sprite
        boolean recentlyJumped // determines jumping cooldown to make jump less spammable
        boolean isJumping  // stores whether the horse is jumping
        double velocity // velocity of the horse
        double acceleration // acceleration of horse
        let sprite // sprite sheet of the horse, animate by css
        constructor(){
            Set keyframe for sprite
            Set animation duration to integer max
            Set own position to 0
            Disable
        }
        function movement()//move expected amount of space over the refresh time//run every
        frame when enabled
                move position to current index +  velocity * refresh time + acceleration/2*(refresh
        time)^2
        }

        function genVelocity ()
        {
                Velocity = velocity + acceleration * refreshTime
        }


        function jump () {
                if (recentlyJumped == true) {
                        end function
                }
                set isJumping to true
                set recentlyJumped to true
                wait 1 second
                set isJumping to false
                wait 0.5 seconds
                set recentlyJumped to false
        }


        function animate(velocity) //sets animation-duration of sprite(css) to 6*velocity,
animation-timing-function to steps(6). Used every frame.



class PlayerHorse extends Horse {
//represents the horse that the player controls; assessing player combos and player jumping is
done here
        let jumpKey // key the player must press to jump (space or defined by constructor)

        let prevKey // previous key that the player pressed, determines if pattern is correct

        array pattern [...] // pattern that keys must be pressed in order to maintain player
        acceleration

        int accelToAdd // updated by changeVelo
        int patternPos // stage in the pattern (if pattern is left-right, then patternPos = 0 when
        player must press left, and 1 when player must press right)

        constructor (array pattern, let jumpKey) {
                sets pattern to the pattern inputted in this constructor
                sets jumpKey to the jumpKey inputted here
                // these are usually the same, but change when multiplayer mode is engaged
        }

        void keyPressed(let pressedKey) {
                // called when player presses a key
                let pressedKey // stores the key that the player pressed to call this function
                let patternKey = pattern[patternPos] // the key that the player is meant to press
                based on their position in the pattern
                if the player pressed the jump key:
                        call jump()
                        end function
                if the player pressed the next key in the pattern (pressedKey == patternKey):
                        add 0.05 to accelToAdd
                        add 1 to patternPos
                if the player pressed the wrong key (not the next key in pattern):
                        set acceleration to 0
                        subtract 1 from velocity, capped at 0 // you have to press the right key
        }
        //when you combo correctly, accelToAdd is incremented to increase the player's speed,
        and this increment is added to the playerHorse's velocity every frame
        function double updateAcel {
                // this function is called every frame, and updates acceleration
                acceleration = accelToAdd + acceleration
                set accelToAdd to zero
        }


        let final image // default and only image that playerHorses can have
}

class RobotHorse extends Horse { // horses that players do not control
        isJumping = true; // dodges all obstacles including Alex's frightening gaze
        2D array possibleNames [][] // first array is the current level, second array is the name
        /*pool of names for level
                1:Rafael, Donnatello, Leonardo, Shelly, Mr. Green, Tuck, Franklin, Michaelangelo
                2:Dessert, Humpy, Dehydrated, Lawrence, Mohammad(probablyshouldntbeused)
                3:Cookie, Miss Zebra, Savannah, Speedy, Jordan
                4:Dash
        1D array of max velocity values for differing levels  maxVelocities[] // later levels have
        higher max velocities which makes roboHorses faster

        constructor (int levelNum) {
                velocity = 0;
                run genName()
                run changeSprite(levelNum)
                run maxVeloSet
        }

        function genAccel (int levelNum) // run this function every time frame is updated
        {
                if velocity < maxVelocities[levelNum]
                generate a random number between 0.2-0.8 * level number
                And set acceleration to that value.
        }
        function genName ()
        {
                Get possibleNames[level# index] and then get a random number from 0 to the
        length, return the randomly chosen name // exclusionary remove the name after being
        chosen from the array
        }


        function changeSprite (int levelNum) // generates image for automated horse upon level
creation
        {
           Sprite is set to the respective sprite sheet for the level (1:turtles, 2:camels, 3:zebra,
4:horse(demon)
        }
}
```