```
/**
 * @file ctc_train.js
 * @author Joey Damico
 * @date September 25, 2019
 * @brief CTC Controller Class for a Clock for the trains
 */

// Import the Custom Clock Class
import Clock from '../Trains/clock.js';

/**
 * CLASS Train
 * @brief Class that keeps track of the time since the start of the
application
 *
 * @details This class is used to keep track and calculate how much
time has passed since the launch
 * of the program, it is used to keep the trains moving at the correct
times
 *
 * MEMBER VARIABLES
 * start_time -> The the games was started
 */
class Train {
    /**
     * constructor()
     * @brief The constructor for the Train class
     *
     * @details This will initialize all the member variables when the
program is started
     *
     * @param p_symbol -> The Train's Symbol
     * @param p_location -> The Trains Inital Location
     * @param p_direction -> The Direction the train is traveling
     * @param p_block_size -> The size of the trains inital block
     */
    constructor(p_symbol, p_location, p_direction, p_block_size) {
        this.clock = new Clock();
        this.clock.startClock();

        this.symbol = p_symbol;
        this.current_location = p_location;
        this.direction = p_direction;
        this.block_size = p_block_size;
        this.block_start = this.clock.getTimeFromStart();

        this.route = true;
    }
    // ---- END constructor() ----
```

```
    /**
     * get_symbol()
     * @brief Getter for the trains symbol
     *
     * @returns The train symbol
     */
    get_symbol() {
        return this.symbol;
    }
    // ---- END get_symbol() ----

    /**
     * update_location()
     * @brief Take in a new location and sets it for the train
     */
    update_location(new_next_location) {
        this.current_location = new_next_location;
        this.block_start = this.clock.getTimeFromStart();
    }
    // ---- END update_location() ----

    /**
     * can_update_location()
     * @brief Determines if the train can move to the next location
     */
    can_update_location() {
        // If The train has a route
        if (this.route) {
            // Check if the train has spent enough time in the curent
block
            if (this.clock.getTimeFromStart() - this.block_start >
this.block_size) {
                return true;
            }
            else {
                return false;
            }
        }
    }
    // ---- END can_update_location() ----

    /**
     * get_location()
     * @brief Getter for the current_location variable
     */
    get_location() {
        return this.current_location;
    }
    // ---- END get_location() ----
```

```
    /**
     * get_block_size()
     * @brief Getter for the block_size variable
     */
    get_block_size() {
        return this.block_size;
    }
    // ---- END get_block_size() ----

    /**
     * set_block_size()
     * @brief Takes in the new block size, and sets the member
variable
     * @param n_size, the new size of the next block
     */
    set_block_size(n_size) {
        this.block_size = n_size;
    }
    // ---- END set_block_size() ----

    /**
     * get_direction()
     * @brief Getter for the direction member variable
     */
    get_direction() {
        return this.direction;
    }
    // ---- END get_direction() ----

    /**
     * get_route()
     * @brief Getter for the route member variable
     */
    get_route() {
        return this.route;
    }
    // ---- END get_route() ----

    /**
     * set_route()
     * @brief Takes in the next route and sets the member variable
     * @param n_route, the trains new route
     */
    set_route(n_route) {
        this.route = n_route;
    }
    // ---- END set_route() ----
}

// Export the panel to be drawn on the screen
```

```
export default Train;
```