BBM104 – PROGRAMMING ASSIGNMENT 4

PROBLEM DEFINITION

In this assignment, we are expected to develop a two-dimensional *"infinite-runner"* car race game using JavaFX framework. Game would consist of cars with randomly generated positions, collision detection, and an animation speed of which would increase as the gamer passes levels. Another problem is that when randomly generating positions for rival cars, a car could overlap another car, or the cars could be placed in a way that it would be impossible for player to pass them: All of which would be unrealistic.

SOLUTION APPROACH

To overcome the situation of which the cars would overlap each other or wouldn't give way to player, I have used (with reluctancy) a brute-force like technique that generates new random positions when a car's position doesn't meet the requirements. To represent every renderable game object, I've developed an abstract *Sprite* class with Image, position and velocity attributes; and to render it, I've used *GraphicsContext* attribute of the *Canvas* object in *Scene*. Also, I have implemented a pause-resume feature (using ESC key).

EXPLANATION PER CLASS

I created three packages and a resource folder with each having its similar files, named "game", "physics", "resources" and "util":

1. **Package "game"**
    a. **Game**: Contains an *ArrayList* of *Sprite* objects, and a render method which renders and updates each game object.
    b. **Package "mechanics"**:
        i. **CarNavigation**: To be able to bypass the delay when pressed a key on the keyboard, it contains 4 boolean attributes which handles the car's movements.
    c. **Package "objects"**:
        i. **Car**: An abstract sprite class representing a car containing position and velocity.
        ii. **RivalCar**: An implementation of *Car*, for rival cars.
        iii. **RedCar**: An implementation of *Car*, for player's car, including *CarNavigation* attribute.
        iv. **Road**: A *Sprite* for representing road, contains road image.
        v. **Sprite**: Base abstract class for all renderable game objects with image, size, position and velocity attributes.
    d. **Package "resources"**:
        i. **AudioPlayer**: A wrapper class for *MediaPlayer*, manages audio files.
        ii. **ResourceManager**: A singleton for managing resources like images and sounds, includes *HashMap*s for each type of resource.
2. **Package "physics"**
    a. **Vector2D**: Contains x and y coordinates and related functions.
3. **Package "util"**
    a. **Constants**: A final class containing constant variables for classes to use.
    b. **Util**: A final utility class containing utility methods like drawing text to canvas or getting a random integer.