Programming Fundamentals II
Lab 7 (Java)

In this lab we are going to improve on our Super Mario Kart game. For this lab we will mainly focus on adding the following features:

1. Use the Kart class in our game.
2. Add the ability for the *kart* to rotate.
3. Add the ability for the *kart* to move forward.
4. Add a new Actor object to represent the finish line.

In this lab we are going to use the following programming features to develop our game:

- **Variables**: to store two types of data:
  1) values such as integers, floats, and strings.
  2) sprites (Kart or Actor).
- **Object Oriented Programming**: we will use our own defined class *Kart*
  1) Briefly modify the class definition
  2) Create an object (class instance), and access its properties and behavior.
- **Boolean logic** to allow us to represent game logic, or the ability to ask questions.
- **If statements**: to provide our code the capability to choose between one or more options.

1 USING THE KART CLASS

In lab 1, we used the generic *Actor* class to represent the kart. In this lab we are going to use our *Kart* class that we implemented in Lab 6. Currently, our code will not run due to errors in Game.java. Let us fix these errors by doing the following:

EXERCISE I:

1. In the Kart class (Actor.java), add a new Boolean property called *gasOn*. The Kart's constructor should initialize this property to false.
2. In the Kart class (Actor.java), add a new Boolean property called *breakOn*. The Kart's constructor should initialize this property to false.
3. In Game.java, we declared the *kart* object of type *Actor* class. Change the type of *kart* from *Actor* to *Kart*.
4. In Game.java init() function, we initialized the *kart* object using the constructor of the *Actor* class. Use instead the constructor of the Kart class. The object should have a *name* as "Mario", *image* as "mario" acceleration as 1, maximum speed as 30, and initial rank as 3.

5. At this time, our program should have no errors. Run the program, and make sure it is working properly.

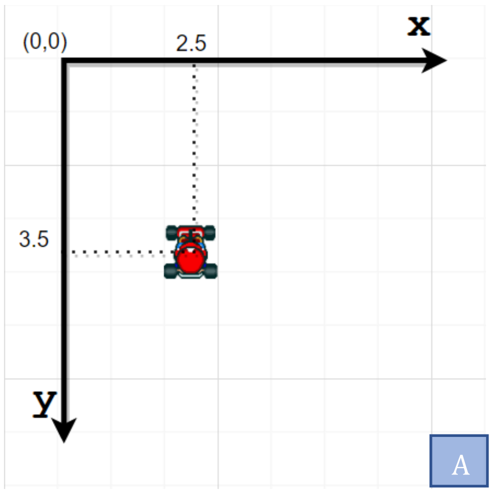
2 ADDING MOVEMENT

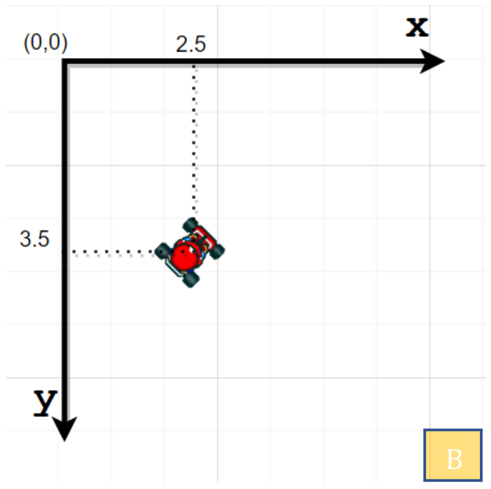
An important feature of race games is the ability to drive the kart. There are different ways to implement character movement in a game. In our game we want the kart to move forward in the direction that the kart is facing. One solution to this problem is to directly increment the kart's *y* position by a fixed value every time the up key is pressed. Although this solution is very simple to program, it will not work when an object rotates. Let us see that through the following exercise:

EXERCISE II:

Suppose we have 7x7 grid:

1. How do we move the kart one block forward in figure A?
2. How do we move the kart one block forward in figure B?





The problem with the previous solution is sometimes we need to modify *x* on its own, *y* on its own or both *x* and *y* values depending on the kart's angle. A better solution is to use the speed and the angle of the kart to calculate the change *dx*, and *dy*. We want the player to control the speed of the cart by pressing the up/down buttons while left/right will rotate its angle.

In Lab 1 we already wrote the logic to handle the ←, →, ↑, or ↓ buttons. We used two Boolean variables:

- gasOn: when this variable is true, the speed should increase until it reaches the maximum speed.
- breakOn: when this variable is true, the speed should decrease until it reaches 0.

As long as the cart has speed, it will always move forward.

EXERCISE III:

Let us first implement the ability for the kart to rotate:

1. Modify the *on_key_down* function so that *da* becomes -1 when → is pressed, and 1 when ← is pressed.
2. Modify the *on_key_up* function so that *da* becomes 0 when ← or → is pressed.
3. In the *update*() function, increment the kart's angle by adding to it *da*.

EXERCISE IV:

Let us next implement the ability for the kart to move forward:

4. We need to modify the kart's *speed* when the gas is on or the break is on. Modify the *update* function so the kart's *speed* is incremented when the gas is on, and decremented when the break is on.
5. Inside the *update* function, call the *move_forward* function. The function *move_forward* takes two arguments: the speed, and angle.
6. The *move_forward* function is incomplete. We need to change the location of the kart. Improve on the *move_forward* function by modifying the kart's position. You need to use the two variables: *dx* (represents the change in *x*) and *dy* (represents the change in *y*).
7. In order to simulate correct movement, we also need to move the *track* when the *kart* moves. In the *move_forward* function, change the *x* and *y* positions of the *track*. The *track* should move 20 times faster than the *kart*. Should the *track* move in the same direction as the *kart*?

3 MANAGING THE LAPS

We want to create a new actor that represents the finish line. We will use this actor in the next lab to keep track of how many laps the *kart* completed.

EXERCISE V:

1. Declare a new Actor called *finishLine*.
2. Using the *Actor* constructor, initialize *finishLine* in the *init*() function. The *x* position should be 205, and *y* should be 345.
3. Draw *finishLine* on the screen.
4. Change this actor's position when the *kart* moves.

4 WHAT TO SUBMIT

Submit your project electronically through D2L by attaching and submitting your Java program files (**Game.java** and **Actor.java**). You do not need to submit any other file.