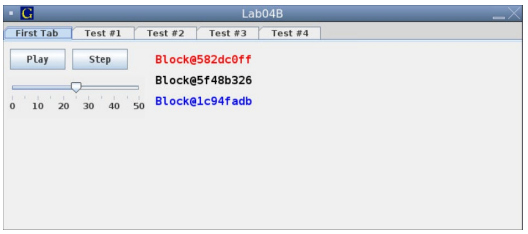
AP Computer Science A
Lab 04B

Name_____
date _____ / _____

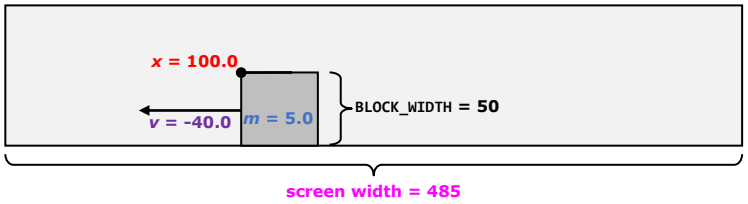
This lab will model square blocks that move along one dimension without friction, and with completely elastic collisions. This is not the *Havok* physics engine- expect some glitches when the blocks collide and as the speed of the blocks increase!

1. Open the **Lab04B** project in the Teams section of **replit**.

   When you press the ▸ Run button, you should see the app shown at right.



2. Each **Block** has the following **double** values: a **mass**, an ***x*-value**, a **velocity**, and a value that represents the **width of the screen** the block lives in.



   A) Add appropriate instance variables to the **Block** class.

   B) In the Java files I wrote I will construct blocks like this:

   This an int...

```
Block b = new Block(5.0, 100.0, -40.0, 485).
```

   Write the **constructor** for the **Block** class.

   C) Finally, add this line of code just before (or after) your instance variables from part (A).

```
private static final int BLOCK_WIDTH = 50;
```

   The keyword **static** indicates that all future blocks will ***share*** this constant (i.e.: all blocks will have the same width). It would be "memory wasteful" for each individual block to have their own separate copy of this variable, which is why it is declared **static** instead of an instance variable.

3. Include these *accessor* methods in the **Block** class:

   This method returns **BLOCK_WIDTH**

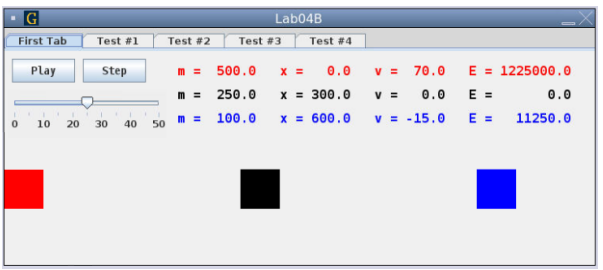
   A) Write **getMass()**, **getVelocity()**, **getX()**, and **public int getBlockWidth()** methods.

   B) Write **getEnergy()**. The (kinetic) energy of a block is ½·mass·velocity^2.

   C) Write (i.e.: override **Object**'s) the **toString()** method. Use the formatting shown below (replace **a**, **b**, **c**, with your instance variables representing the mass, *x*-value, and velocity).

```
String.format("m = %6.1f   x =%6.1f   v =%6.1f   E =%10.1f", a, b, c, this.getEnergy())
```

   Your app should look like this:

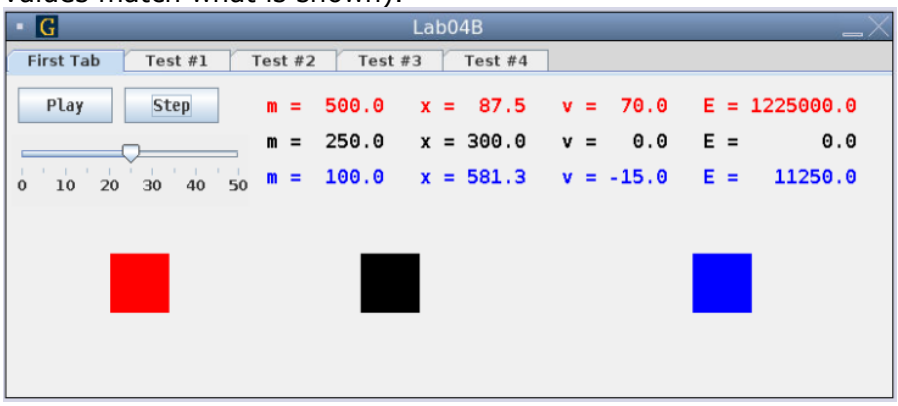


4. In this step you will write the *mutator* method **`public void move(double timeIncrement)`**

   - Increase the *x*-value of the block by the product of its velocity and the **`timeIncrement`**.
   - If the edge of a block reaches (or surpasses) either edge of the screen (recall that the screen width was the last explicit parameter in the **`Block`** constructor), make the block's velocity opposite in sign (elastic collisions preserve kinetic energy).

   When you press the **Step** button the red square should move to the right and the blue square should move to the left.

   The screenshot shown below is the result of pressing the **Step** button five times (check that your *x*-values match what is shown).



   When you press the **Play** button the blocks will pass through each other (to be fixed in the next step), but they should bounce off the left and right edges of the app.

5. Write the *mutator* method **`public void checkCollision(Block other)`**.

   - Check to see whether the two blocks (**`this`** and **`other`**) are intersecting or meeting. This would normally be a complicated check since collisions can occur from the left side or right side, but since all blocks have equal width you can check the **distance** between any pair of corresponding points on the blocks. You could also use the **`intersects`** method of the **`Rectangle`** class.
   - If the two blocks do intersect, an elastic collision occurs. The equations of the new velocities are

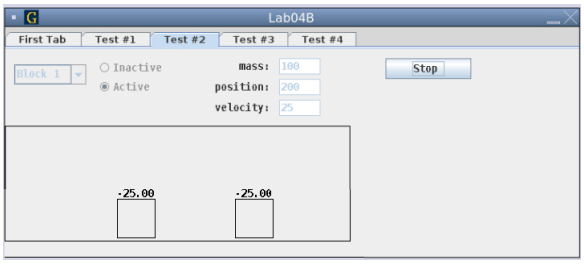
$$v_1' = \frac{v_1(m_1 - m_2) + 2m_2 v_2}{m_1 + m_2} \quad \text{and} \quad v_2' = \frac{v_2(m_2 - m_1) + 2m_1 v_1}{m_1 + m_2}.$$

   Note that v_1 and v_2 are the **old** (prior to the collision) velocities of the 1st and 2nd block!

   Make sure your blocks bounce off each other.

6. Finally, check that the tabs labeled Test#1 – Test#4 are working by pressing the **Start** button for each test.

   Make sure that the blocks bounce off each other and that the blocks do not leave the rectangle they live in.



7. Once you have finished, click the ✓Submit button near the top right of your browser. This notifies me that you think your lab assignment is complete and ready for grading.