**Report**

Jugglr – a juggling game

Use the mouse to move the paddle. See how long you can keep the balls in the air!

**Build instructions**

IDE - Intellij Idea Community Edition

I used Maven to import the following libraries:
org.jbox2d:jbox2d-library:2.2.1.12 - physics library, ported from C++ library Box2D
http://mvnrepository.com/artifact/org.jbox2d/jbox2d-library/2.2.1.1

org.slf4j:slf4j-jdk14:1.7.72 (JBox2D dependency)
http://mvnrepository.com/artifact/org.slf4j/slf4j-jdk14/1.7.7

I found it easiest to use Intellij's import library feature to do this (File->Project Structure->Libraries->New Project Library->From Maven)

## Classes

### Jugglr.java

This class is responsible for creating the Menu and Display classes. I initially had the Menu class as an inner class of this one, in order to register an ActionListener on it. I decided that this wasn't very object-oriented, however. I ended up separating the Menu to its own class and passing the ActionListener to each button in that class as a reference. This way I was able to preserve the object-oriented nature of the program while still being able to have the Jugglr class listen to events in the Menu. When an game-starting action is detected, the menu window is disposed and the game window is created.

### Menu.java

I was having trouble setting the layout/width of the JButtons in the Menu class (only one button was being displayed), so I decided to use a GridBagLayout instead. I found this to be a very effective way of setting up a simple menu for my purposes.

### Display.java

This class paints objects from the Physics class on the screen, in addition to an end of game message, as well as handling mouse movements. I decided it was logical for the Display to contain the Physics object as it would need to frequently get information from this class in order to paint it. The variable REPAINT_SPEED determines the frequency that repaint() is called (60fps in this case).

One of the main issues I had with this class was converting data from JBox2D objects (e.g. Vec2) to data which could be interpreted by Java.awt methods such as fillOval(). I had to write a couple of helper functions to accomplish this. Likewise, scaling from metric units, as well as working with floats (and trying to prevent lossy conversion/overflow) was tricky.

As this class developed I found that it was becoming somewhat unwieldy, and was handling things it shouldn't be handling (like setting the paddle position – I moved this function to the Physics class). After refactoring, however, things looked a bit more sensible.

**Physics.java**

This class contains the JBox2D world object, as well as separate ArrayLists for each Entity type (Ball and Wall), and a unique Wall object representing the platform. I initially had just one ArrayList for all Entity objects, however this led to an issue in the Display paintComponent() method – I was having to use instanceof on each object in the list to determine its type before calling the relevant draw method. I believe this is a relatively expensive operation, so I separated each type out into its own list.

Physics also contains a score variable - I felt that although the game score is not directly related to the Physics class, it is dependent on the update frequency, so I left it there.

Performance – I was also concerned about performance issues, as there are a lot of calculations to be done at each world step and I wanted to minimise the amount of work done on the Event Dispatch Thread. I found the best solution (for my purposes) was to have a TimerTask inner class in the Physics class, thereby allowing the physics calculations to happen on a separate thread (at a specified interval).

**Entity.java**

This is an abstract class representing all the information for a JBox2D world object. The constructor creates a BodyDef object and sets its position, and leaves the rest up to the concrete sub-class to set up. The only issue here is that each Entity needs a reference to the JBox2D World which contains it – a cyclic dependency. Since this is only used to construct the object, however, and the Physics object can be used to perform all meaningful operations on the World, I felt that this was acceptable.

**Ball.java/Wall.java**

These classes both handle JBox2D information, with the main differences being the type of Shape and Body/Fixture properties (density, friction, restitution, BodyType) contained in each. I was planning to have a Box class (similar to a Ball), but this involved too much complexity in converting vertices for the Display/calculating angles, so I abandoned it.

**RunningFlag.java**

I needed a way to keep track of whether the game is in progress. It became evident that a simple boolean variable in the Physics class would not work – how does it relay a change in this variable to the Display class? I ended up creating a separate RunningFlag class which extends Observable, and having Display implement the Observer interface to keep track of it.

It did, however, feel a bit unnecessary creating a class just to hold a variable.

**Testing**

I conducted testing in each main class, testing constructor methods for Ball and Wall objects, as well as Physics and Display where possible.

**Reflections**

If I was designing the game again, I would probably have the main Jugglr class handle the main wiring up of the program i.e. containing the Display, Menu, Physics and RunningFlag objects and connecting them/passing information around as appropriate.

I also found that Jbox2D and Swing/awt don't get on particularly well, mainly due to the former's use of floats. Next time I would probably choose either a physics library that is more friendly to integers or a graphics library that is better for floating point numbers. By the same token, it might have been useful to have a separate Utility class for checking/converting these values.

**Known bugs/issues**

**ArrayIndexOutOfBoundsException –** happens at unpredictable times, does not seem to have any effect on the game flow. Seems to be caused by the setPlatformPos() method.

Output:

```
java.lang.ArrayIndexOutOfBoundsException: -1
	at
org.jbox2d.collision.broadphase.DynamicTree$TreeNodeStack.pop(DynamicTree.java:899)
	at org.jbox2d.collision.broadphase.DynamicTree.query(DynamicTree.java:168)
	at org.jbox2d.collision.broadphase.BroadPhase.updatePairs(BroadPhase.java:178)
	at org.jbox2d.dynamics.ContactManager.findNewContacts(ContactManager.java:173)
	at org.jbox2d.dynamics.Body.setTransform(Body.java:333)
	at Display.setPlatformPos(Display.java:127)
	at Display.access$200(Display.java:14)
	at Display$Mouser.mouseMoved(Display.java:157)
	at java.awt.Component.processMouseMotionEvent(Component.java:6580)
	at javax.swing.JComponent.processMouseMotionEvent(JComponent.java:3342)
	at java.awt.Component.processEvent(Component.java:6304)
	at java.awt.Container.processEvent(Container.java:2236)
	at java.awt.Component.dispatchEventImpl(Component.java:4891)
	at java.awt.Container.dispatchEventImpl(Container.java:2294)
	at java.awt.Component.dispatchEvent(Component.java:4713)
	at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4888)
	at java.awt.LightweightDispatcher.processMouseEvent(Container.java:4538)
	at java.awt.LightweightDispatcher.dispatchEvent(Container.java:4466)
	at java.awt.Container.dispatchEventImpl(Container.java:2280)
	at java.awt.Window.dispatchEventImpl(Window.java:2750)
	at java.awt.Component.dispatchEvent(Component.java:4713)
	at java.awt.EventQueue.dispatchEventImpl(EventQueue.java:758)
	at java.awt.EventQueue.access$500(EventQueue.java:97)
	at java.awt.EventQueue$3.run(EventQueue.java:709)
	at java.awt.EventQueue$3.run(EventQueue.java:703)
	at java.security.AccessController.doPrivileged(Native Method)
	at
java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomai
n.java:76)
	at
java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomai
n.java:86)
	at java.awt.EventQueue$4.run(EventQueue.java:731)
	at java.awt.EventQueue$4.run(EventQueue.java:729)
	at java.security.AccessController.doPrivileged(Native Method)
	at 
java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomai
n.java:76)
	at java.awt.EventQueue.dispatchEvent(EventQueue.java:728)
	at java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:201)
	at java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:116)
	at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:105)
	at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
	at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:93)
	at java.awt.EventDispatchThread.run(EventDispatchThread.java:82)
java.lang.ArrayIndexOutOfBoundsException: -2
	at 
org.jbox2d.collision.broadphase.DynamicTree$TreeNodeStack.pop(DynamicTree.java:899)
	at org.jbox2d.collision.broadphase.DynamicTree.query(DynamicTree.java:168)
	at org.jbox2d.collision.broadphase.BroadPhase.updatePairs(BroadPhase.java:178)
	at org.jbox2d.dynamics.ContactManager.findNewContacts(ContactManager.java:173)
	at org.jbox2d.dynamics.World.solve(World.java:1117)
	at org.jbox2d.dynamics.World.step(World.java:598)
	at Physics$UpdateTask.run(Physics.java:126)
	at java.util.TimerThread.mainLoop(Timer.java:555)
	at java.util.TimerThread.run(Timer.java:505)
```

**NullPointerException** – seems to be caused by world.step(), the World can't find its objects? I'd like to think this is an issue with JBox2D...

Output:

```
java.lang.NullPointerException
	at org.jbox2d.collision.broadphase.DynamicTree.removeLeaf(DynamicTree.java:650)
	at org.jbox2d.collision.broadphase.DynamicTree.moveProxy(DynamicTree.java:121)
	at org.jbox2d.collision.broadphase.BroadPhase.moveProxy(BroadPhase.java:109)
	at org.jbox2d.dynamics.Fixture.synchronize(Fixture.java:439)
	at org.jbox2d.dynamics.Body.synchronizeFixtures(Body.java:1128)
	at org.jbox2d.dynamics.World.solve(World.java:1113)
	at org.jbox2d.dynamics.World.step(World.java:598)
	at Physics$UpdateTask.run(Physics.java:124)
	at java.util.TimerThread.mainLoop(Timer.java:555)
	at java.util.TimerThread.run(Timer.java:505)
java.lang.NullPointerException
	at org.jbox2d.collision.broadphase.DynamicTree.removeLeaf(DynamicTree.java:650)
	at org.jbox2d.collision.broadphase.DynamicTree.moveProxy(DynamicTree.java:121)
	at org.jbox2d.collision.broadphase.BroadPhase.moveProxy(BroadPhase.java:109)
	at org.jbox2d.dynamics.Fixture.synchronize(Fixture.java:439)
	at org.jbox2d.dynamics.Body.synchronizeFixtures(Body.java:1128)
	at org.jbox2d.dynamics.World.solve(World.java:1113)
	at org.jbox2d.dynamics.World.step(World.java:598)
	at Physics$UpdateTask.run(Physics.java:124)
	at java.util.TimerThread.mainLoop(Timer.java:555)
```

```
        at java.util.TimerThread.run(Timer.java:505)
```

Full disclosure: sometimes the game freezes with no explanation.