Springies Design Document

- Design goals
    - Clear separation of functionality (i.e. user interactions in listener classes, environment and force interactions in EnvironmentManager, force calculations in forces class, object behaviors in object classes). This separation is defined by our separate classes and packages and ensures that we know where interactions are being manipulated. Thus, if we have to debug a function related to a specific behavior, we know where to start first.
    - Clear hierarchy between related objects to eliminate redundant code and ensure that functionality is consistent across related objects. This way, a modification of a class of objects' behavior is simple and only needs to be applied once to the superclass enveloping all the classes.
    - Write and clearly name methods according to their function so it is easy to go back through and understand what our code does.
    - Allow code to be extensible and easily debugged
- How to add new features
    - New user interactions:
        - Simply add the keystroke/mouse movement to the appropriate listener and call the interaction through the EnvironmentManager or Springies
        - i.e.: in OnKeyListener, add a case to the switch() statement and write its functionality into EnvironmentManager
    - New objects:
        - Create the object class extending PhysicalObject to define how it is drawn and how it interacts with the JGame and JBox worlds.
        - Define how it moves and how to apply world forces to the objects
        - Create all other necessary behaviors and appropriately call behaviors from EnvironmentManager
        - Add to Assembly as necessary
    - New XML specifications:
        - Create new parser class extending XMLParser, and define the appropriate behaviors according to different nodes which the parser will encounter
    - New forces:
        - Create new force implementing the Force class and show how forces will interact with Mass objects (any object in the JBox world that has mass) and will interact with the environment
        - Call the appropriate force in EnvironmentManager so force will be applied to all objects
    - New interactions:
        - Create behavior in EnvironmentManager
    - Change in JGame specifications
        - We did not have to do a lot of this so currently the JGame specifications are in main and springies but can be refactored out to make a cleaner code
- Major design choices
    - Using SAX XML Parser
        - We've used SAX in other projects before, but never other XML parser
        - Pro: simple to use, don't need to worry about calling next line, SAX iterates through nodes and will process nodes as the parser reaches each node

        - Cons: have to either store each node in new data structure (redundant) or build model/environment as the parser reads the file
    - Building assembly right in ModelParser
        - Pro: Avoid creating redundant data structure/constructors (information read from file would have to be stored somewhere before creation)
        - Con: Assumes that each XML document specifies only 1 assembly
        - Con: Did not separate by functionality
    - Using JGame built-in listeners rather than building our own
        - Pro: JGame has already configured listeners to record data every frame of JGame
        - Pro: Don't need to worry about debugging the actual registering of keystrokes/mouse movements, just have to worry about response to keystrokes
        - Cons: Have to call clearLastKey() after every frame
        - Cons: If JGame is buggy, we are subject to all those bugs
    - Choosing to make Forces an interface
        - Pro: Ensure that all forces have a method (calculateForce) that can be called by EnvironmentManager and will return the same object (Vec2) that can be applied to objects with masses
        - Pro: Restricts Mass calculation to objects with masses
- Assumptions
    - Assumed that springs and muscles will never have masses
    - Assumed that XML files will only ever be formatted as in the examples given
    - Assumed that once assemblies are created, they are set -- would require some additional methods to edit assemblies after making the assembly object
    - Assumed that directions are given in radians – how Java calculates on default – this will be a problem because the direction in environment.xml is given in degrees.