1. Write a new class, loan2. Inherit the ReferAssignGetSet class to handle the application-independent properties of referencing, assignment, get and set. This is available at the website. See the Inheritance lecture notes for a reminder of how this will work. In the zip file, there are a few more ppt (and pdf) slides, which give the details of the final implementation.

Your class should have the following properties

- .Principal initial loan amount
- .InterestRate annual interest rate, 6.0 means 6%
- .MonthlyPayment monthly payment
- .Duration duration of loan, in months
- .DerivedProperty which (of above 4) is computed from the others
- .Balance See below
- .Equity See below

Some facts about the properties

(a) All properties are gettable.

(b) The DerivedProperty is always settable. Its possible values are the character strings Principal, InterestRate, MonthlyPayment, and Duration. The user need only type part of the string, and need not use upper case. Hence if A is a lone2 object, the user can type

```plaintext
>> A.DerivedProperty = 'm' % or 'mont' or 'Mo', etc
```

to change the property to MonthlyPayment.

(c) The property described by the value of DerivedProperty is not settable (it is the derived property). The other 3 properties are settable.

(d) The property described by the value of DerivedProperty is always updated to be the correct value, based on the calculation provided by loancalc.m.

(e) The Balance field is only gettable. It is a column vector, of length Duration. The k’th entry is the balance after the k’th month. Using the notation from the previous assignment, this is is simple $p_k$. The last entry of Balance should be 0.

(f) The Equity field is only gettable. It is a column vector, of length Duration. The k’th entry is the equity after the k’th month. Using the notation from above, this is

$$E_k = L - p_k$$

The last entry of Equity should be the loan amount.
2. Write a new class, `loangui2`. It will be a gui, of almost identical functionality to the `loangui.m`. However, `loangui2` should inherit the `hgtoolkit` class so that it behaves as an object, can hide data, set listeners, and has a get/set interface to changes its properties. Leverage the `loan2` class - one (critical) piece of data in the private database of the class will be a `loan2` object. Although vastly oversimplified, the `e177gui2` class has similar features (using `hgtoolkit`).

(a) The properties of `loangui2` can be exactly the same as the properties of the `loan2` objects.

(b) When the user interacts with the graphical tool, the properties change, and `get` reveals these changes. The properties can also be changed directly with `set` and any such changes are also reflected in the graphical tool.

(c) A user can set listeners for any of the (settable) properties. Remember that this service is automatically provided by the `hgtoolkit` implementation. However, you should check that this works, which gives some positive indication that you are using the `hgtoolkit` properly.

(d) Finally, use two `hvpatch` objects to make the Principal and Duration draggable, from the graph.

Check at the HW section of the website for additional hints/corrections.