# Adding an IBMFunction to a DisMELS life stage class

1. Create a public class that extends AbstractIBMFunction (or implements IBMFunctionInterface)
   1. optional: if extending AbstractIBMFunction, have the class implement a “category” class like IBMGrowthFunctionInterface that extends IBMFunctionInterface and indicates the category of the function.
2. Use the Netbeans’ BeanInfoEditor to create a BeanInfo class for the IBMFunction created in 1.
   1. Using the BeanInfoEditor “Designer” tab
      1. EXCEPT for functionName, click each property listed under “Properties” and uncheck “Include in BeanInfo” to exclude the property from the BeanInfo.
      2. Right click each event source listed under “Event Sources” and uncheck “Include in BeanInfo” to exclude it from the BeanInfo.
      3. Right click the following methods listed under “Methods” and uncheck “Include in BeanInfo” to exclude them from the BeanInfo:
         1. calculate
         2. clone
         3. createInstance
         4. getParameterDescription
         5. setParameterDescription
   2. In the getBdescriptor method of this BeanInfo, add the folllowing line:

beanDescriptor.setValue("persistenceDelegate",new AbstractIBMFunctionPersistenceDelegate());

* 1. Save file.

3. For each life stage class (e.g., LifeStageClass) that will utilize the function, add the function to the mapOfPotentialFunctions for the appropriate function category in the method createMapToSelectedFunctions for the class that describes the parameters for the life stage (e.g., LifeStageClassParameters). Thus, if GrowthFunction1 and GrowthFunction2 are the desired growth functions to make available, then the part of the method createMapToSelectedFunctions in LifeStageClassParameters dealing with growth functions might look like

//add the FCAT\_growth category to the set of function category keys for this class

setOfFunctionCategories.add(FCAT\_Growth);

//create the map from function categories to potential functions in each category

String cat; Map<String,IBMFunctionInterface> mapOfPotentialFunctions; IBMFunctionInterface ifi;

cat = FCAT\_Growth;

mapOfPotentialFunctions = new LinkedHashMap<>(2);

mapOfPotentialFunctionsByCategory.put(cat,mapOfPotentialFunctions);

ifi = new GrowthFunction1(); mapOfPotentialFunctions.put(ifi.getFunctionName(),ifi);

ifi = new GrowthFunction2(); mapOfPotentialFunctions.put(ifi.getFunctionName(),ifi);

where FCAT\_growth is a class level String identifying the function category (=”growth”, e.g.), setOfFunctionCategories is an inherited class-level set containing all function categories, and mapOfPotentialFunctionsByCategory is an inherited class-level map giving the mapping from the category name (as a String) to the mapOfPotentialFunctions for that category.

4. In the associated life stage class (e.g., LifeStageClass), make the selected IBMFunction available using something like

fcnGrowth = params.getSelectedIBMFunctionForCategory(MaleImmatureParameters.FCAT\_Growth);

in the setParameters method, where fcnGrowth will be the IBM function selected in the parameters file and params is an instance of LifeStageClassParameters (continuing the example started in 3. above). Then, call fcnGrowth with the appropriate values where its needed in the step method of LifeStageClass.

## Fully-classified list of classes

wts.models.DisMELS.framework.IBMFunctions.AbstractIBMFunction

wts.models.DisMELS.framework.IBMFunctions.IBMFunctionInterface

wts.models.DisMELS.framework.IBMFunctions.IBMGrowthFunctionInterface