**M4 Borg Design Pattern**

Hamid Shahrestani Mehr - 9729747

**Factory Pattern**

**Intro**:

According to Wikipedia “Factory is a creational design pattern to implement the concept of factories and deals with the problem of creating objects (products) without specifying the exact class of object that will be created. The essence of this pattern is to "Define an interface for creating an object, but let the classes that implement the interface decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses.”

In BorgCalendar, GridBagConstraints class is used numerously in many classes. The GridBagConstraints class specifies constraints for components that are laid out using the GridBagLayout class. To facilitate instantiation of objects from this class, a factory class is designed. This class is used 39 times within UI packages to instantiate GridBagConstraints objects. GridBagConstraints object can be created in several ways with different fields populated. Factory method encapsulates the process of creation and saves lots of time.

To capture the following interactions I used Object Aid UML tool.

**Code Snippets**

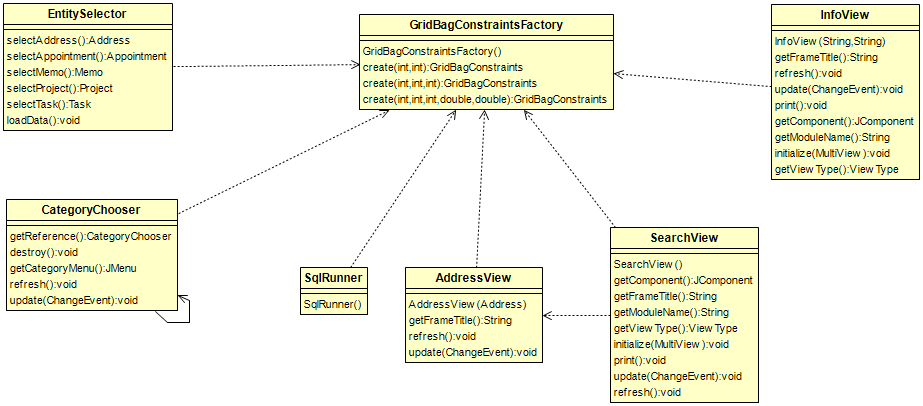
For example in the following code snippet, JPanel has an add method that takes GridBagConstaint object as an argument. To instantiate objects of this type, a factory class is used.

|  |
| --- |
| private JPanel getTopPanel() {  if (topPanel == null) {  topPanel = new JPanel();  topPanel.setLayout(new GridBagLayout());  topPanel.add(buttonPanel, GridBagConstraintsFactory.create(0, 1,  GridBagConstraints.BOTH, 1.0, 0.0));  topPanel.add(getJScrollPane(), GridBagConstraintsFactory.create(0,  0, GridBagConstraints.BOTH, 1.0, 1.0));  }  return topPanel;  } |

|  |
| --- |
| public class GridBagConstraintsFactory {  static private final Insets defaultInsets = new Insets(4, 4, 4, 4);  public static GridBagConstraints create(int x, int y) {  GridBagConstraints gbc = new GridBagConstraints();  gbc.gridx = x;  gbc.gridy = y;  gbc.insets = defaultInsets;  return gbc;  }    public static GridBagConstraints create(int x, int y, int fill) {  GridBagConstraints gbc = create(x, y);  gbc.fill = fill;  return gbc;  }  public static GridBagConstraints create(int x, int y, int fill,  double weightx, double weighty) {  GridBagConstraints gbc = create(x, y, fill);  gbc.weightx = weightx;  gbc.weighty = weighty;  return gbc;  }  } |

**Class Diagram**

For example the picture below displays some of the classes that use this factory class.



**Sources:**

# Design Patterns: Elements of Reusable Object-Oriented Software