# *Programming II (420-B20-HR)*

# *Lab 4 –Abstract Classes; WindowBuilder*

Date assigned: Wednesday, February 10, 2016

Date due: **Wednesday, February 10, 2016**

**Learning Objectives**

1. Learn to declare and use abstract classes and methods.
2. Learn how to design and build a **JFrame** using **WindowBuilder**.
3. Review static class variables and methods.

**To Be Handed In:**

1. The ***username*\_B20\_L04\_Project** folder should be uploaded to **Moodle**.

**To Start:**

1. Download and unzip the folder **B20\_L04\_Project** from **Moodle**. Rename it to ***username*\_B20\_L04\_Project**.
2. Start **Eclipse** and select your **420-B20\Labs** folder as your workspace.
3. Create a new **Java Project** called ***username*\_B20\_L04\_Project**.

# Abstract Classes

***Purpose:*** Learn to declare and use abstract classes and methods.

***To Do:***

## Open the **Product.java** class in the **videoStoreSystem** package.

## Change the **setCategoryCode(String)** method to an abstract method as shown here:

**public abstract void setCategoryCode(String categoryName);**

Compile the **Product** class. What happens? The abstract method setCategoryCode in type Product can only be defined by an abstract class\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Explanation***: If a class contains an abstract method, you must declare the class to be abstract.

## Make the **Product** class abstract by adding the word **abstract** before the word **class** on the class header.

## Change the **getCategory()**, **setFormatCode(String)** and **getFormat()** methods to abstract.

## Open the **Movie** class. Hover over the underlined error. What's the message? The type Movie must implement the inherited abstract method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Explanation***: If a class is abstract, all of its subclasses must be abstract or must implement all the abstract methods.

## Click the red x in the margin beside the **Movie** class header. Double-click **Add unimplemented methods**. Leave the default methods for now.

## Repeat the previous step with the **Game** class.

## ~~Run the~~ **~~AddProductFrame~~** ~~program and try to add a movie and a game.~~

## ~~In the constr~~The type Movie must implement the inherited abstract method~~uctor for the~~ **~~AddProductFrame~~** ~~class, change~~

**~~product = null;~~**

~~to~~

**~~product = new Product();~~**

~~Compile it. What happens? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_~~

***~~Explanation~~***~~: An abstract class cannot be instantiated.~~

# B. Creating a Java Windows application using Google WindowBuilder

***Purpose:*** Learn how to use the Google WindowBuilder plugin to create a Java Windows application in the **Eclipse** IDE.

***To Do:***

We are going to create a frame to add movies and games to the Video Store system. To help us design the frame we are going to use **Google's WindowBuilder**. The first step is to use the visual editor to design and build the frame.

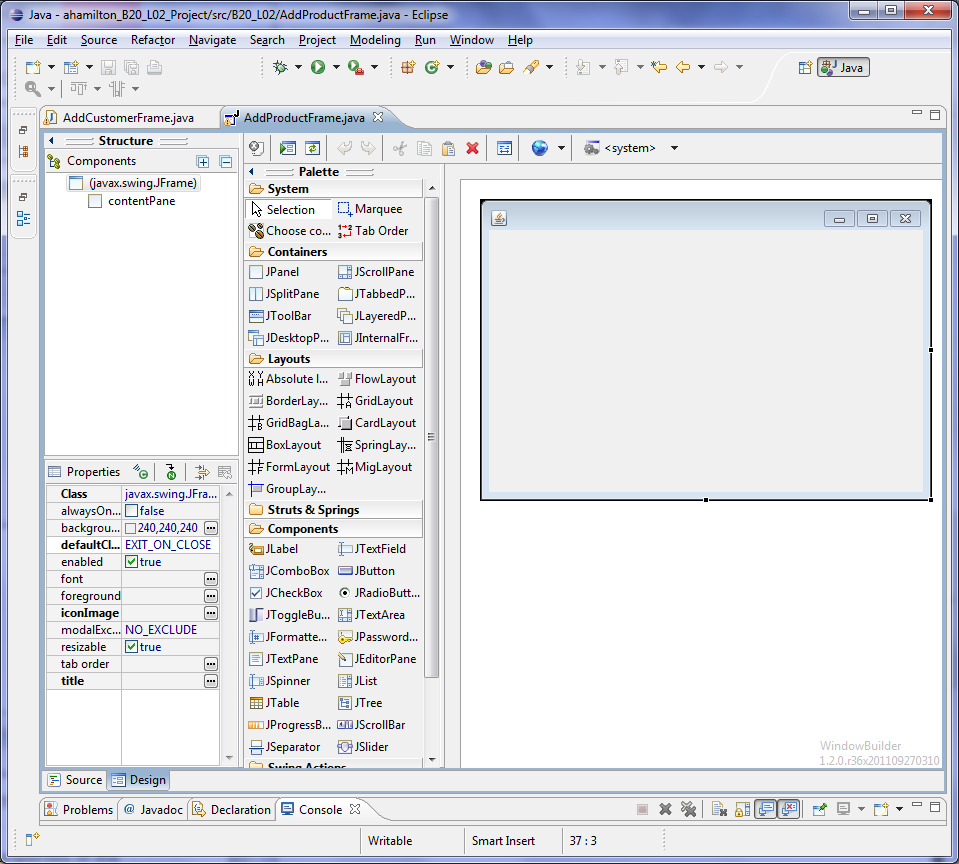
## Create the **Add Product** frame as follows:

### Select **File**🡪 **New** 🡪 **Other** 🡪 **WindowBuilder** 🡪 **Swing** **Designer** 🡪 **JFrame** to start the new **JFrame** wizard.

### Click **Next**. Make sure that you are in your source folder and select the **videoStoreSystem** package. The name of the class should be **AddProductFrame**.

### Click **Finish**. The skeleton of a frame will be created for you.

## Click the **Design** tab at the bottom of the class source pane. This opens the design window:



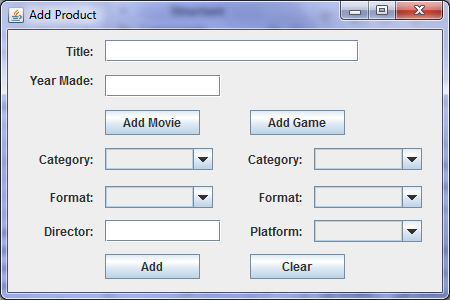
## Click **(javax.swing.JFrame)** in the **Components** panel. The properties of the frame will be displayed in the **Properties** panel below the **Components** panel under **Structure**. Type *Add Product* in the field beside **title** and click **Enter**. Look at the drawing panel. The title has been added.

## Run the frame.

## Click **AbsoluteLayout** in the **Layouts** section of the **Palette** and then click the panel. This allows us to control the position of each of the components we want to add.

## Click **JLabel** under **Components** section of the **Palette**. Position the cursor in the upper left of the JFrame (under the "Add Product" title) in the drawing panel. Click to add the label. Type *Title:* in the label. (Notice that the text property changes correspondingly.) Select **RIGHT** from the drop-box beside **horizontalAlignment** in the **Properties** panel.

## Add labels, textfields, buttons and JComboBoxes to create a frame that looks like:



You can run the frame at any time to see what it looks like.

## The components will be automatically labeled. In some cases (i.e. the labels and buttons) the names are clear. However the names of the textfields and combo boxes are not sufficiently descriptive. Rename them to **fldTitle**, **fldYearMade**, **cmbxMovieCategory**, **cmbxMovieFormat**, **fldDirector**, **cmbxGameCategory**, **cmbxGameFormat** and **cmbxPlatform** as appropriate. You should also rename the labels for the Movie and Game categories and formats to indicate whether it is for a movie or a game. (e.g. lblMovieCategory, lblGameCategory)

## Now we want to add the appropriate lists to the combo boxes. Click on the combo box for the Game format. Click **Model** in the **Properties** panel. Type the following list in the text area under **ComboBox model items:**

DVD

CD

## Run the program to see the combobox. At the moment it shows DVD. To prevent the user from forgetting to select a model type, edit the combobox list to add a blank line as the first line. Run the program again to make sure that it is working.

## Complete the combo boxes for the movie category, movie format, game category, game format and game platform. (See pages 2 and 3 of the case study for the values.) If no default value is given, add a blank line at the beginning to force the user to make a selection.

## Make your labels all the same size. (Click on a label that is the desired size and then CTRL-Click the other labels. Select the *Replicate width* icon () from the frame toolbar.) In a similar way, you can use the other icons to improve the look of your frame.

## Run your frame to see what it looks like. Resize and realign the components if necessary so that all the text in the comboboxes is visible.

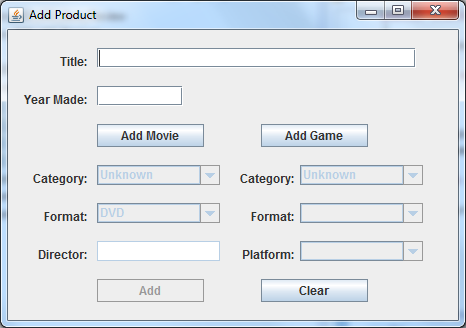
## Right-click the **Clear** button and select **Add event handler** 🡪 **action** 🡪 **actionPerformed**. This creates an **actionPerformed()** method for the **Clear** button and switches to the method in the source frame.

## In the **actionPerformed()** method, add a call to the **clearAllFields()** method.

## Code the private **clearAllFields()** method. It should set the content of all the text fields to "";

## Test your method by running the frame. Enter garbage in each of the fields and then press the **Clear** button.

## When a new product is added, we want to control the fields that user can enter. Initially only the Title Year Made, Add Movie, Add Game and Clear buttons should be enabled. Select the other components and toggle off **enabled** in the **Properties** panel. Your completed frame should look similar to:



**Note:**

If you wish to install WindowBuilder on your own computer, do the following:

To install **WindowBuilder**, go to **Help** -> **Install New Software** and enter

<http://download.eclipse.org/windowbuilder/WB/release/R201506241200-1/4.4/> in the **Work with** box. When the options appear, **Swing Designer** and **WindowBuilder Engine** from list. Click **Next** twice, accept the terms of the license agreement and click **Finish**. **Eclipse** will automatically restart at this point.

# Assignment 1 Design Approval

***Purpose:*** Ensure that the design and test cases for assignment 1 are correct so that you can move on to implementation.

***To Do:***

## Show the teacher your frame design, class diagram and test cases. Ask questions. Discuss. Make corresponding updates.

## With any time you have left, create your assignment 1 frame using WindowBuilder.

# Homework

## Complete the **Week 4 Quiz** on Moodle by Feb. 14.