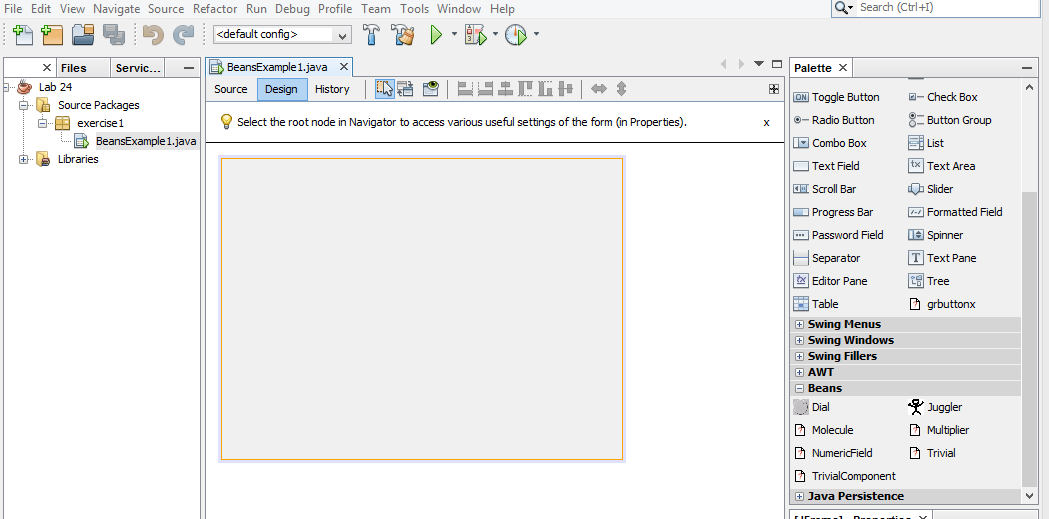
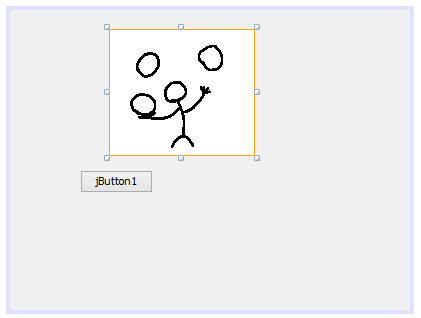
**Lab 24**

Create a new Project called Lab 24. Before we get started, we’ll have to add the example beans used in this chapter to the NetBeans palette. NetBeans installs Java beans in a central location that can be used by multiple projects in the editor.

1. To install our beans, download the sample beans JAR file, magicbeans.jar, from Moodle.
2. Save the file locally and then select Tools → Palette → Swing/AWT Components to add the beans.
3. Click “Add from Jar,” then navigate to and select the magicbeans.jar file and click Next. The wizard then shows you a list of beans that our JAR contains (there should be seven altogether).
4. Select all of them (click the first and shift-click the last), then click Next.
5. NetBeans prompts you for the palette category under which you wish to file these; select Beans (which would otherwise be empty), then click Finish and Close.
6. Next, we need to create a class file to put into our project. Create a new package called exercise1. Then select New File from the File menu. NetBeans prompts you with a wizard. In the Categories pane, select Swing GUI Forms and in the File Types pane, select JFrame Form. Choosing JFrame Form gives us a Java class file extending JFrame with the basic structure of a GUI application already set up for us. Click Next, then give the file a name, such as BeansExercise. Now click Finish and you will see a screen similar to the following:



1. By right-clicking on the JFrame (our top-level container) in either the workspace or the tree, you can select Set Layout to specify the layout manager for the frame. For now, try using AbsoluteLayout, provided by NetBeans. This allows you to arbitrarily place and move beans within the container. You wouldn’t want to do this in general, but for our examples it will make life a bit easier.
2. On the right side of the screen are a Palette pane and a Properties pane, which is currently showing some properties of the currently selected file, but will show the properties of the currently selected bean component when we add one to the design area. The Palette has groupings for different types of beans. The Swing group includes all the standard Swing components. AWT holds older AWT components. Layouts holds Java layout managers.
3. Now scroll down and select Beans to see the beans we imported earlier.
4. Scroll back up to the Sing Controls and click on a button (this will be a JButton) from the Swing Controls group of the palette, and then click in the workspace. When the JButton was first loaded by NetBeans, it was inspected to discover its properties. When we select an instance of the button, NetBeans displays these properties in the Properties pane and allows us to modify them.
5. Scroll down to the Beans palette, select the Juggler bean and place it in the workspace. The animation starts, and the juggler begins juggling some roughly drawn beans, as shown in Figure below.



1. You can see that this bean has a different set of properties, including an interesting one called animationRate. It is an integer property that controls the delay in milliseconds between displays of the juggler’s frames. Try changing its value to 25 and press Enter to see the speed change. Good beans give you immediate feedback when you change their properties. Uncheck the checkbox next to the Boolean juggling property to stop the juggling.
2. Using the Properties pane, change the label of your button to read “Start.”
3. Now click the small Connection Mode icon at the top of the GUI builder (the second icon, showing two items with arrows pointing at one another).
4. After pressing the button, NetBeans is waiting for us to select two components to “hook up.” Click first on the Start button and then on the Juggler. NetBeans pops up the Connection Wizard, indicating the source component (the button) and prompting you to choose from a large list of events. Most of them are standard Swing events that can be generated by any kind of JComponent. What we’re after is the button’s action event.
5. Expand the folder named **action**, and select **actionPerformed** as the source event.
6. At the bottom of the dialog box NetBeans indicates the name of an event handler method that it will generate for us. Leave the method name as is. Click Next to go to the Specify Target Operation screen for the Juggler
7. The wizard prompts us to choose a property to set on the Juggler. The display shows three of the Juggler’s properties. Choose the juggling property as the target and click Next.
8. Enter true in the Value field and click Finish. NetBeans takes you to the source view and shows you the method it has generated to respond to the button action.
9. We have completed a hookup between the button and the Juggler. When the button fires an action event, the juggling property of the Juggler is set to true.
10. Scroll around the source view and take a look at the code that NetBeans has generated to make this connection for us. Specifically, in the initComponents()method of our class, it has created an anonymous inner class to serve as theActionListener for ActionEvents from our button (which it has named jButton1).
11. Click back to Design View and add a second JButton labeled “Stop.” Click the Connection Wizard icon; select the Stop button and the Juggler as its target. Again, choose the actionPerformed method as the source, but this time, instead of selecting a property on the Juggler, click the Method Call radio button to see a list of available methods on the Juggler bean. Scroll all the way down and select the stopJuggling() method. Click Finish to complete the hookup, and look at the generated code if you wish. With this, we have seen an example of hooking up a source of action events to generate an arbitrary method call on a bean. (Of course, there is a startJuggling() method as well, which we could have used for the first button.)
12. Right-click on the *BeansExercise.java* file in the Projects tab of the Explorer pane (or in the source view of the file) and select Run File. Watch as NetBeans compiles and runs our example. You should be able to start and stop the juggler using the buttons! When you are done, quit the juggler application and return to the GUI editor.