In Java, when you add a `WindowListener` to a component using the `addWindowListener` method, you typically pass an instance of a class that implements the `WindowListener` interface as an argument. This class should override the methods defined in the `WindowListener` interface to handle specific window events.

Here are the different ways to pass parameters to the `addWindowListener` method:

1. **Anonymous Inner Class**:
   - You can create an anonymous inner class that implements the `WindowListener` interface and overrides its methods. This allows you to pass parameters to the inner class constructor and use them within the overridden methods.

   ```java
   frame.addWindowListener(new WindowListener() {
       public void windowOpened(WindowEvent e) {
           // Your code here
       }

       // Other overridden methods
   });
   ```

2. **Named Inner Class**:
   - You can create a named inner class that implements the `WindowListener` interface in a separate class file. This class can have instance variables and a constructor to accept parameters.

   ```java
   class MyWindowListener implements WindowListener {
       private String message;

       public MyWindowListener(String message) {
           this.message = message;
       }

       public void windowOpened(WindowEvent e) {
           // Your code here, using the 'message' parameter
       }

       // Other overridden methods
   }

   // Usage:
   frame.addWindowListener(new MyWindowListener("Hello, World!"));
```

```

3. **Lambda Expressions (Java 8 and later)**:
   - If you're using Java 8 or later, you can use lambda expressions to create concise event listeners. However, lambda expressions don't allow you to pass parameters directly, so you would need to use final or effectively final variables from the enclosing scope.

   ```java
   String message = "Hello, World!";
   frame.addWindowListener(new WindowAdapter() {
       public void windowOpened(WindowEvent e) {
           System.out.println(message);
       }
   });
   ```

4. **Instance Variables**:
   - You can use instance variables of your class to pass data to the `WindowListener`. This approach allows you to set the necessary data before adding the listener.

   ```java
   public class MyFrame extends Frame {
       private String message;

       public MyFrame(String message) {
           this.message = message;
           addWindowListener(new MyWindowListener());
       }

       private class MyWindowListener extends WindowAdapter {
           public void windowOpened(WindowEvent e) {
               System.out.println(message);
           }
       }
   }

   // Usage:
   MyFrame frame = new MyFrame("Hello, World!");
   frame.setSize(300, 200);
   frame.setVisible(true);
   ```
   ```

Choose the method that best suits your application's requirements and code organization. Lambda expressions are more concise but may have limitations regarding parameter passing. Inner classes and instance variables provide more flexibility for passing parameters.