实验三：软件需求的跟踪分析

选定的开源 IDE 项目： IDEA。

实验过程：

明确提出需求 R 的文本，获取需求 R 的有关讨论文本；

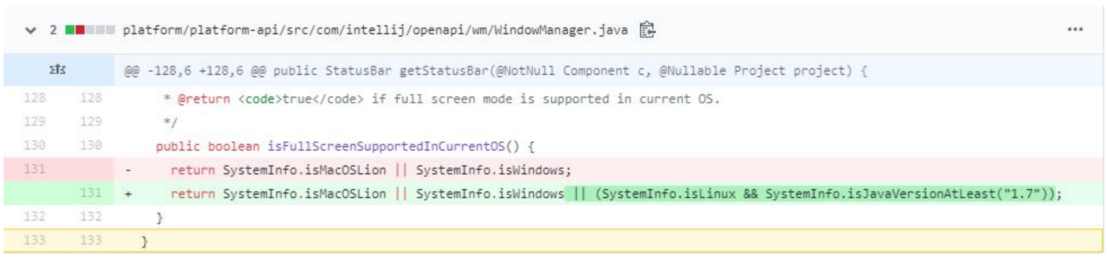
需求 R 提出： Unable to toggle fullscreen view on linux。

项目地址: https://github.com/JetBrains/intellij-community/pull/69

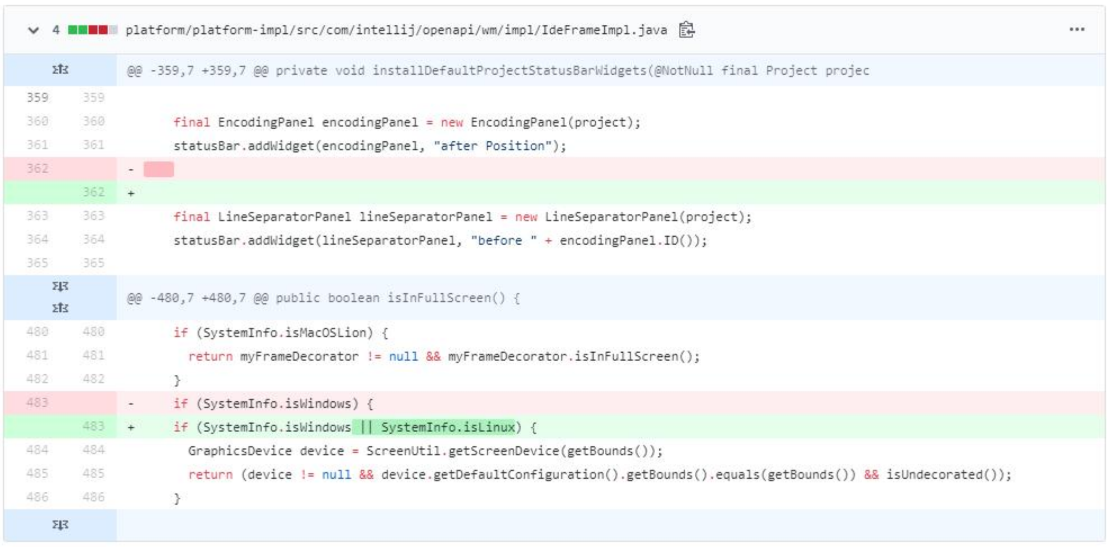
相关文本：

Draft of supporting of FullScreen Mode at Linux. Checking on JRE 1.7, 1.8 XFCE with SawFish WM.

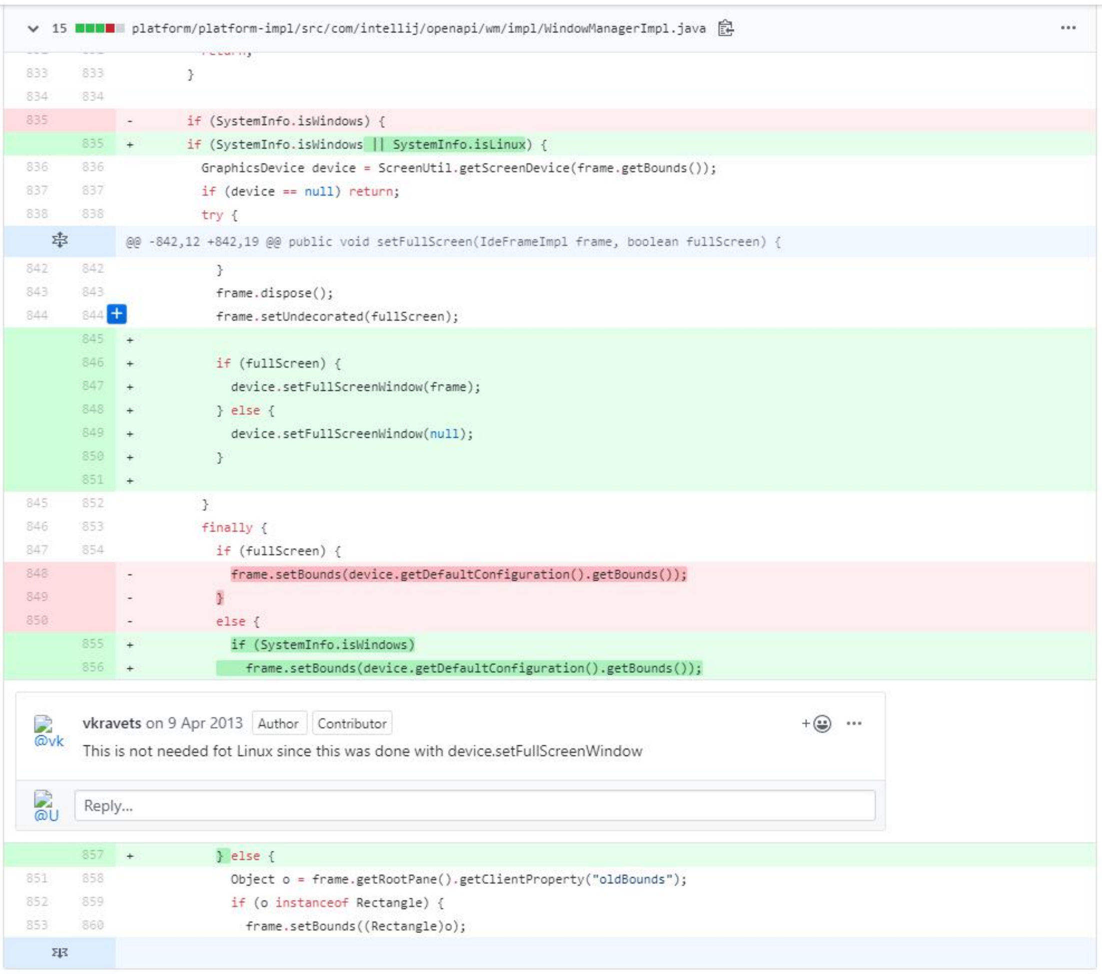
实现需求的代码：



```
2 platform/platform-api/src/com/intellij/openapi/wm/WindowManager.java

@@ -128,6 +128,6 @@ public StatusBar getStatusBar(@NotNull Component c, @Nullable Project project) {
      * @return <code>true</code> if full screen mode is supported in current OS.
      */
    public boolean isFullScreenSupportedInCurrentOS() {
-     return SystemInfo.isMacOSLion || SystemInfo.isWindows;
+     return SystemInfo.isMacOSLion || SystemInfo.isWindows || (SystemInfo.isLinux && SystemInfo.isJavaVersionAtLeast("1.7"));
    }
  }
```



```
4 platform/platform-impl/src/com/intellij/openapi/wm/impl/IdeFrameImpl.java

@@ -359,7 +359,7 @@ private void installDefaultProjectStatusBarWidgets(@NotNull final Project projec

      final EncodingPanel encodingPanel = new EncodingPanel(project);
      statusBar.addWidget(encodingPanel, "after Position");
-
+
      final LineSeparatorPanel lineSeparatorPanel = new LineSeparatorPanel(project);
      statusBar.addWidget(lineSeparatorPanel, "before " + encodingPanel.ID());

@@ -480,7 +480,7 @@ public boolean isInFullScreen() {
      if (SystemInfo.isMacOSLion) {
        return myFrameDecorator != null && myFrameDecorator.isInFullScreen();
      }
-     if (SystemInfo.isWindows) {
+     if (SystemInfo.isWindows || SystemInfo.isLinux) {
        GraphicsDevice device = ScreenUtil.getScreenDevice(getBounds());
        return (device != null && device.getDefaultConfiguration().getBounds().equals(getBounds()) && isUndecorated());
      }
```



15 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java

```diff
         }
 
-        if (SystemInfo.isWindows) {
+        if (SystemInfo.isWindows || SystemInfo.isLinux) {
           GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
           if (device == null) return;
           try {
@@ -842,12 +842,19 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
             }
             frame.dispose();
             frame.setUndecorated(fullScreen);
+
+            if (fullScreen) {
+              device.setFullScreenWindow(frame);
+            } else {
+              device.setFullScreenWindow(null);
+            }
+
           }
           finally {
             if (fullScreen) {
-              frame.setBounds(device.getDefaultConfiguration().getBounds());
-            }
-            else {
+              if (SystemInfo.isWindows)
+                frame.setBounds(device.getDefaultConfiguration().getBounds());
```

vkravets on 9 Apr 2013 Author Contributor

This is not needed fot Linux since this was done with device.setFullScreenWindow

Reply...

```diff
+            } else {
               Object o = frame.getRootPane().getClientProperty("oldBounds");
               if (o instanceof Rectangle) {
                 frame.setBounds((Rectangle)o);
```

可以看出代码主要是增加了对操作系统的判断和处理

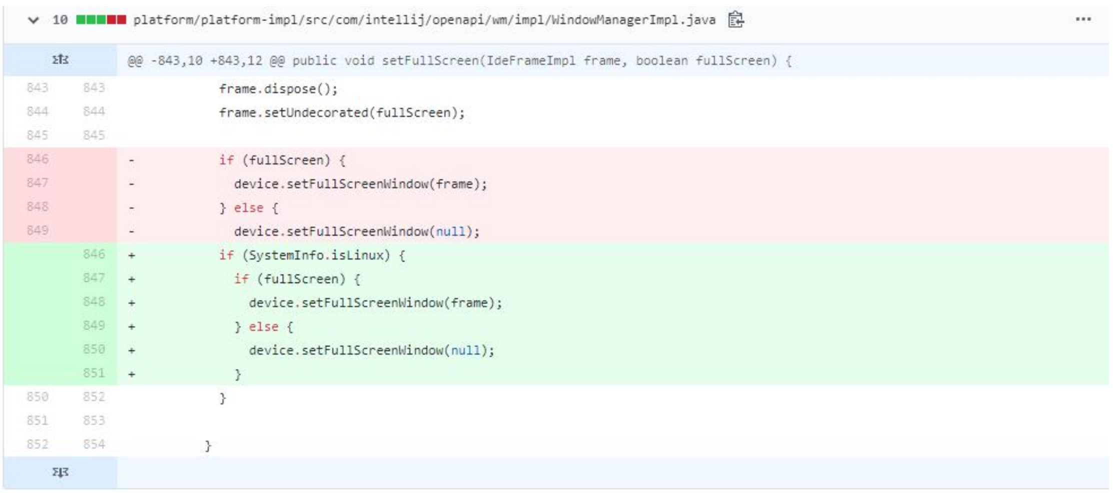
同时作者表示在 java1.6 下存在该全屏问题，仅在 java1.7 版本后支持全屏，并且并没有能力在 Gnome3/2 或 KDE 桌面引擎下检测。

文本：

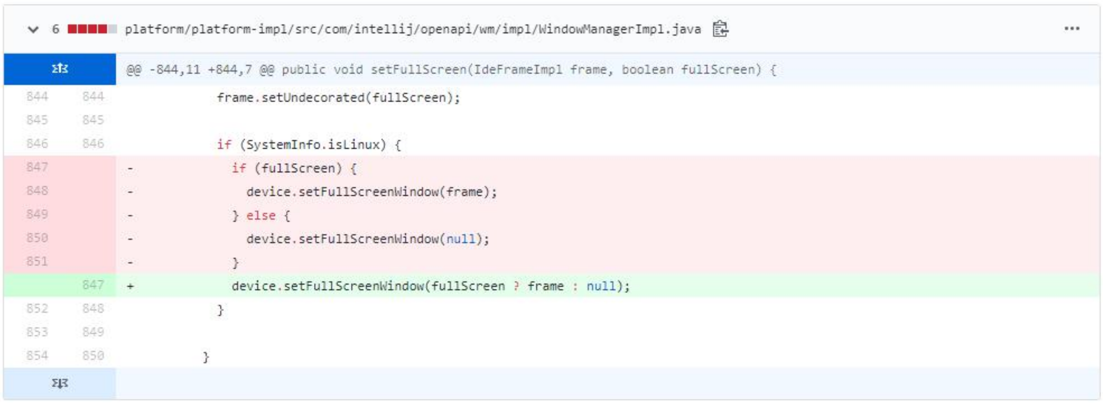
Since there is some issue with full screen in 1.6 of Java... Supporting only from 1.7... which is checks in the code...

I don't have ability to check in Gnome3/2 or KDE...

同时作者对之前全屏相关代码两次次修改，对系统进行检测（第二次主要为了缩短代码量）



```
10 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -843,10 +843,12 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
            frame.dispose();
            frame.setUndecorated(fullScreen);

-           if (fullScreen) {
-             device.setFullScreenWindow(frame);
-           } else {
-             device.setFullScreenWindow(null);
+           if (SystemInfo.isLinux) {
+             if (fullScreen) {
+               device.setFullScreenWindow(frame);
+             } else {
+               device.setFullScreenWindow(null);
+             }
            }

          }
```



```
6 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -844,11 +844,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
            frame.setUndecorated(fullScreen);

            if (SystemInfo.isLinux) {
-             if (fullScreen) {
-               device.setFullScreenWindow(frame);
-             } else {
-               device.setFullScreenWindow(null);
-             }
+             device.setFullScreenWindow(fullScreen ? frame : null);
            }

          }
```

之后有人回复表示必须在所有桌面管理引擎上进行 check，作者回复会在 VM 上测试然后给出了测试结果

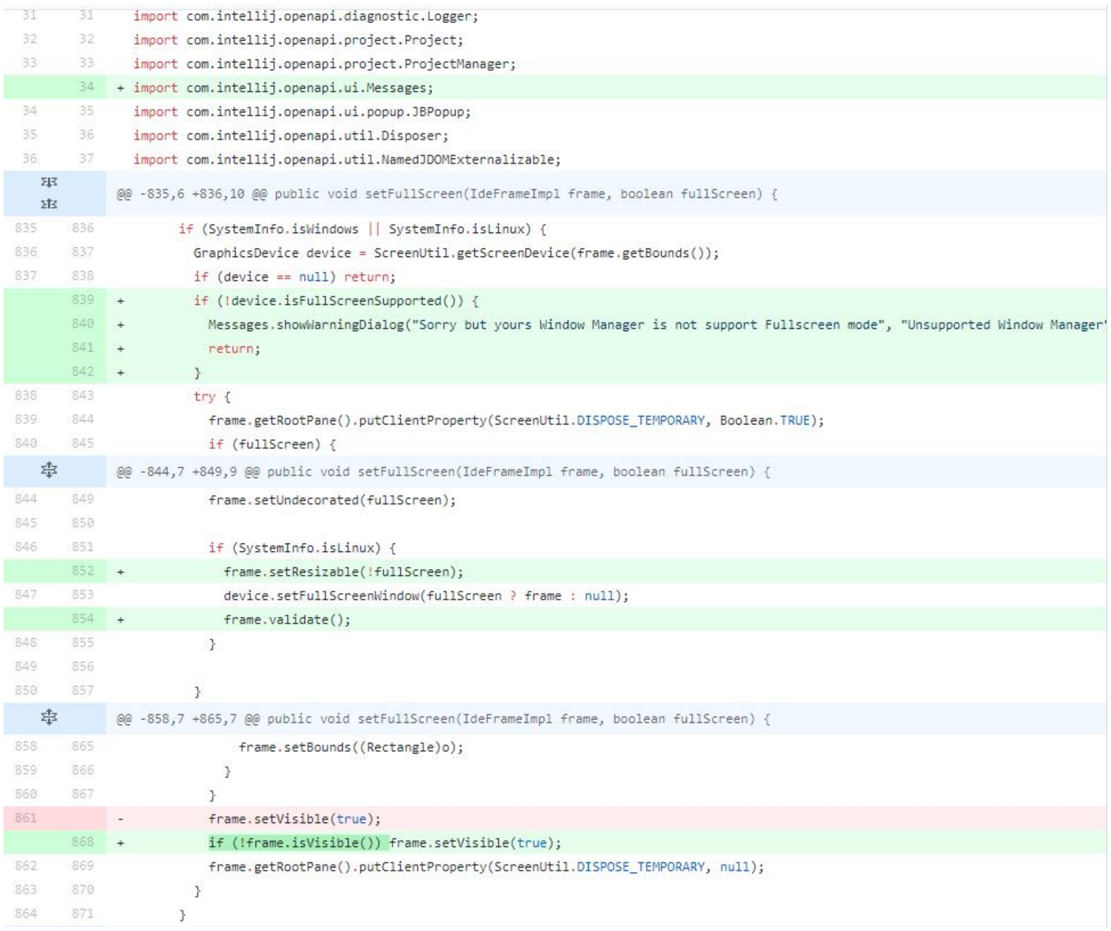
Phanteon (Elementary Desktop/OS) - Passed
KDE - Passed
Gnome Classic - don't working at all =(((It seems issue with compozite manager, need to check on the latest version
Gnome Classic (no effect) - Passed
Gnome Shell - Passed (issue with top bar depth, after switch (alt-tab) or Active IDEA menu e.g. Alt+F everything is ok)
XFCE - Passed with the same issue with Gnome Shell (after switching (alt-tab) or activating IDEA menu e.g.Alt-F works well)
Ubuntu Unity - Passed. works well if window is not maximized.

Thus I think in most of cases this code working well, but need to think about some tweaks for Unity and XFCE and Gnome Shell...

Any suggestion how I can check which WM is used?

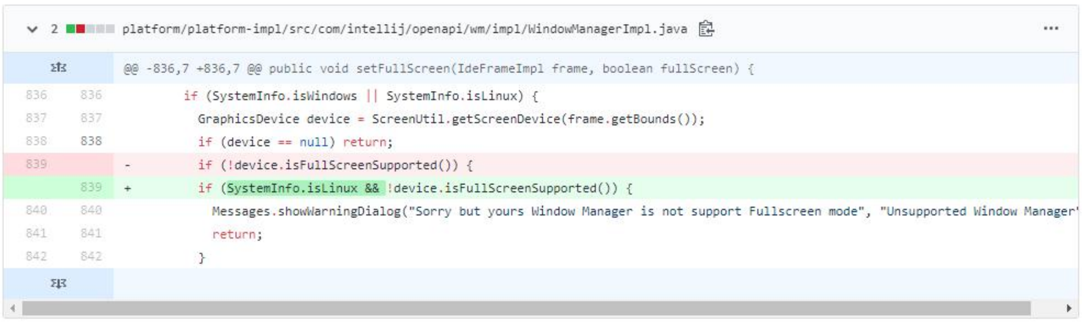
之后又给出了 Gnome Classic 上不工作的原因
Problem with Gnome Classic is found. Related to Compiz.... To be able to have full screen

support in Compiz need to turn on Legacy Fullscreen Support in Workaraund settings.
并且修改代码对不支持的 WM 给出提示信息



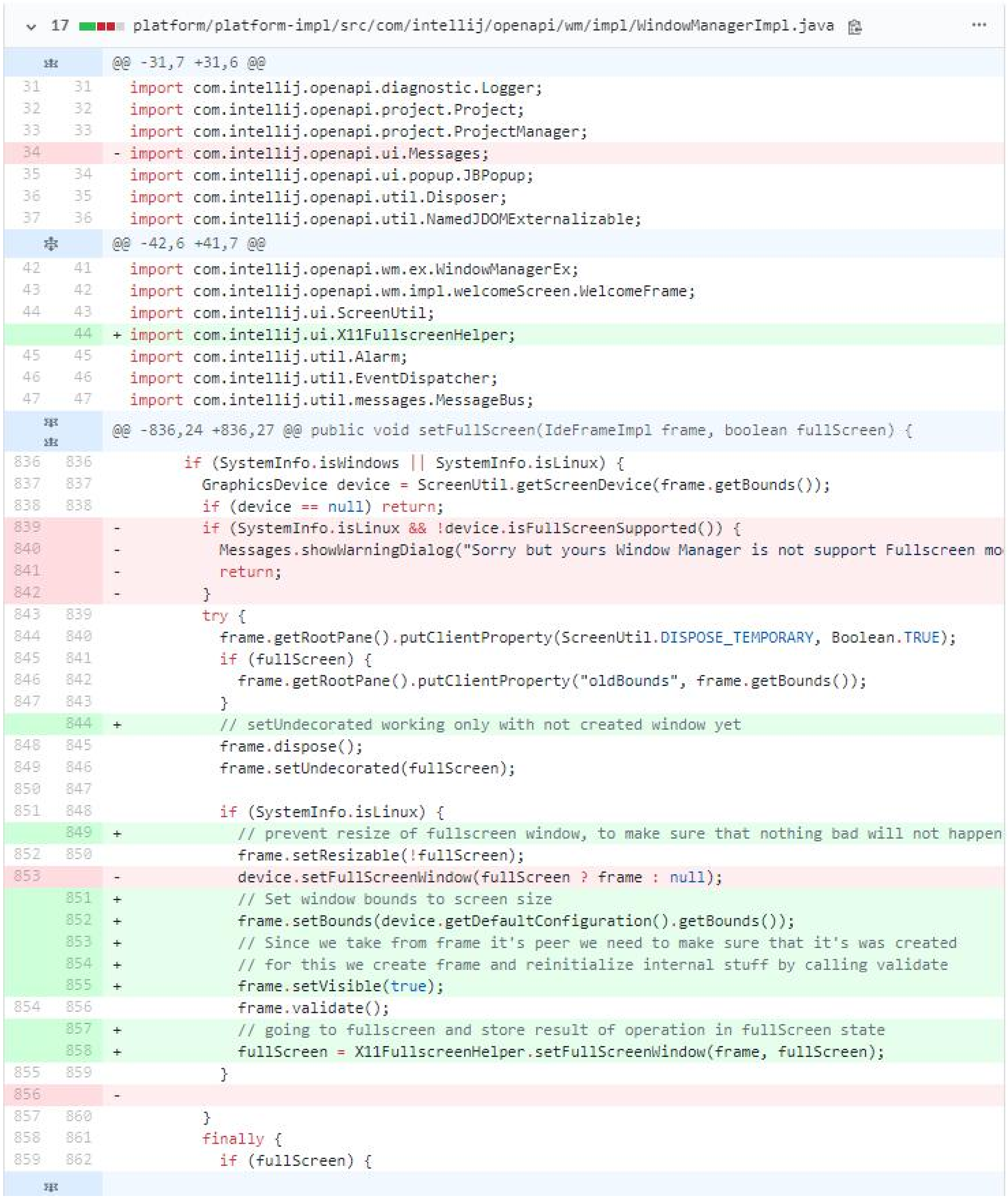
```
 import com.intellij.openapi.diagnostic.Logger;
 import com.intellij.openapi.project.Project;
 import com.intellij.openapi.project.ProjectManager;
+import com.intellij.openapi.ui.Messages;
 import com.intellij.openapi.ui.popup.JBPopup;
 import com.intellij.openapi.util.Disposer;
 import com.intellij.openapi.util.NamedJDOMExternalizable;
@@ -835,6 +836,10 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
       if (SystemInfo.isWindows || SystemInfo.isLinux) {
         GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
         if (device == null) return;
+        if (!device.isFullScreenSupported()) {
+          Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mode", "Unsupported Window Manager'
+          return;
+        }
         try {
           frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, Boolean.TRUE);
           if (fullScreen) {
@@ -844,7 +849,9 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
           frame.setUndecorated(fullScreen);

           if (SystemInfo.isLinux) {
+            frame.setResizable(!fullScreen);
             device.setFullScreenWindow(fullScreen ? frame : null);
+            frame.validate();
           }

         }
@@ -858,7 +865,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
               frame.setBounds((Rectangle)o);
             }
           }
-          frame.setVisible(true);
+          if (!frame.isVisible()) frame.setVisible(true);
           frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, null);
         }
       }
```

之后又对该段代码再次修改仅对 Linux 系统进行消息提示



```
2 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -836,7 +836,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
       if (SystemInfo.isWindows || SystemInfo.isLinux) {
         GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
         if (device == null) return;
-        if (!device.isFullScreenSupported()) {
+        if (SystemInfo.isLinux && !device.isFullScreenSupported()) {
           Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mode", "Unsupported Window Manager'
           return;
         }
```

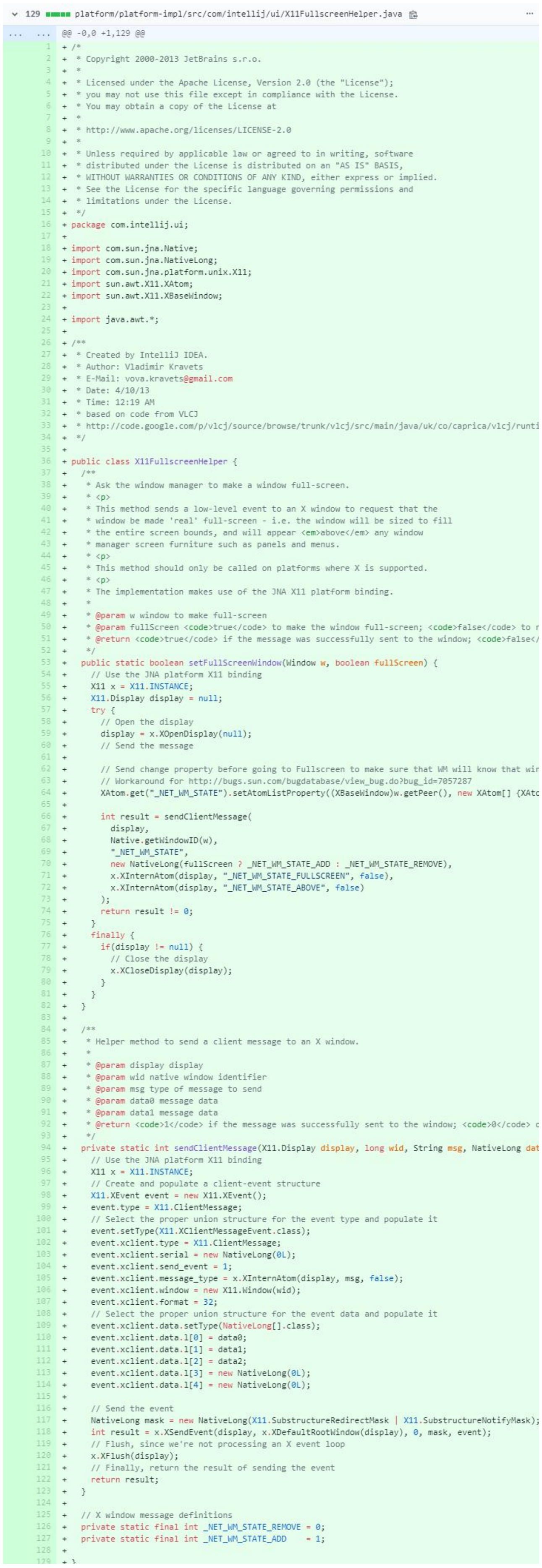
之后又修改了全屏模式的方法，使用了原生的 X11 调用



```
17  platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java

@@ -31,7 +31,6 @@
   import com.intellij.openapi.diagnostic.Logger;
   import com.intellij.openapi.project.Project;
   import com.intellij.openapi.project.ProjectManager;
 - import com.intellij.openapi.ui.Messages;
   import com.intellij.openapi.ui.popup.JBPopup;
   import com.intellij.openapi.util.Disposer;
   import com.intellij.openapi.util.NamedJDOMExternalizable;
@@ -42,6 +41,7 @@
   import com.intellij.openapi.wm.ex.WindowManagerEx;
   import com.intellij.openapi.wm.impl.welcomeScreen.WelcomeFrame;
   import com.intellij.ui.ScreenUtil;
 + import com.intellij.ui.X11FullscreenHelper;
   import com.intellij.util.Alarm;
   import com.intellij.util.EventDispatcher;
   import com.intellij.util.messages.MessageBus;
@@ -836,24 +836,27 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
         if (SystemInfo.isWindows || SystemInfo.isLinux) {
           GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
           if (device == null) return;
 -         if (SystemInfo.isLinux && !device.isFullScreenSupported()) {
 -           Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mo
 -           return;
 -         }
           try {
             frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, Boolean.TRUE);
             if (fullScreen) {
               frame.getRootPane().putClientProperty("oldBounds", frame.getBounds());
             }
 +           // setUndecorated working only with not created window yet
             frame.dispose();
             frame.setUndecorated(fullScreen);

             if (SystemInfo.isLinux) {
 +             // prevent resize of fullscreen window, to make sure that nothing bad will not happen
               frame.setResizable(!fullScreen);
 -             device.setFullScreenWindow(fullScreen ? frame : null);
 +             // Set window bounds to screen size
 +             frame.setBounds(device.getDefaultConfiguration().getBounds());
 +             // Since we take from frame it's peer we need to make sure that it's was created
 +             // for this we create frame and reinitialize internal stuff by calling validate
 +             frame.setVisible(true);
               frame.validate();
 +             // going to fullscreen and store result of operation in fullScreen state
 +             fullScreen = X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
             }
 -
           }
           finally {
             if (fullScreen) {
```

增加新文件 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java 使用 X11 调用实现全屏。

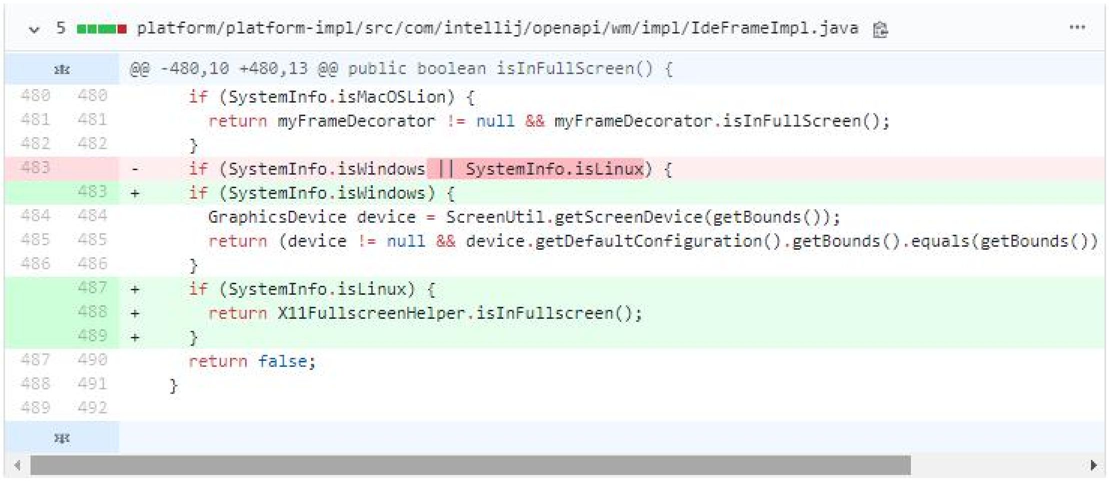
新代码:



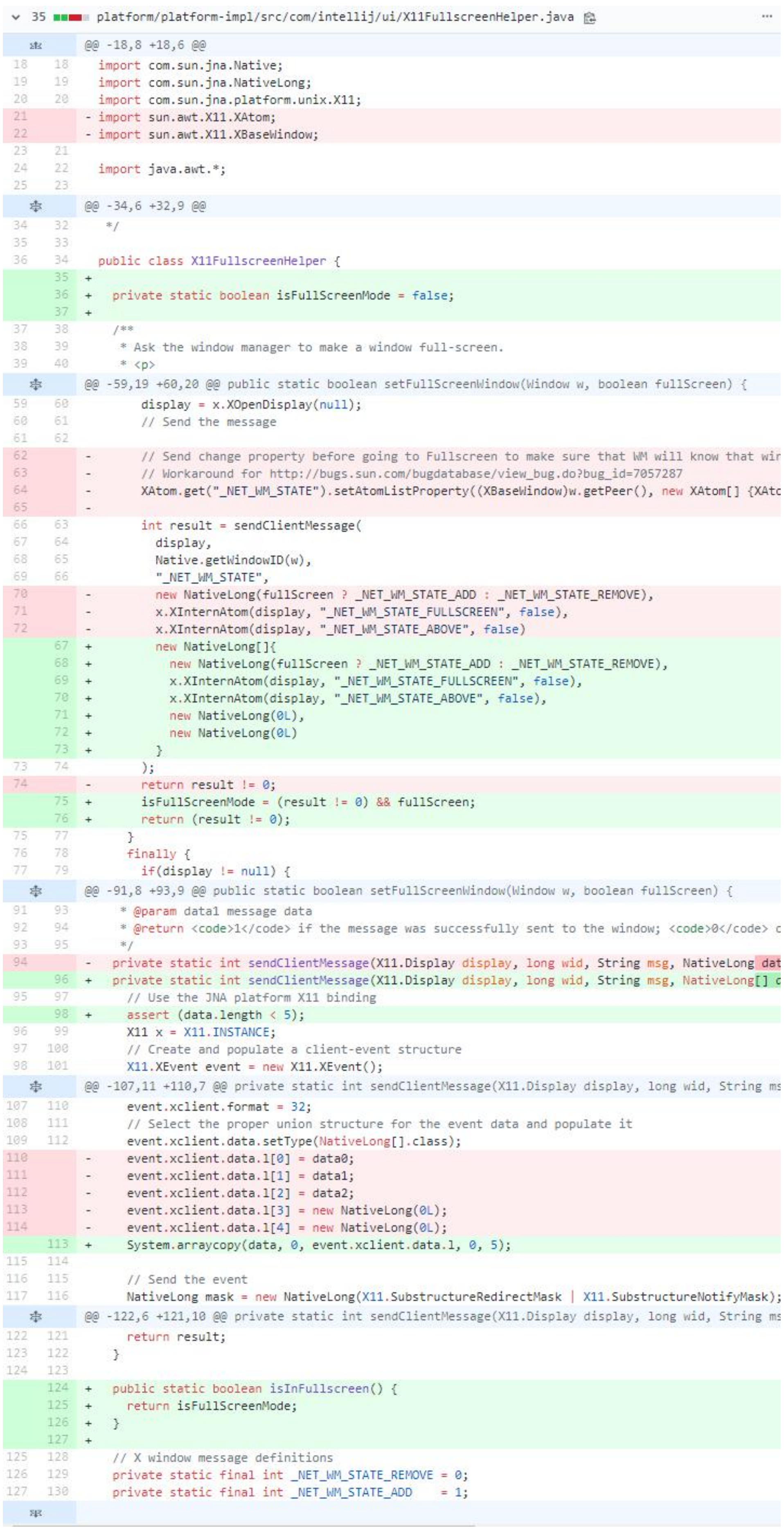
129 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java

```diff
@@ -0,0 +1,129 @@
+ /*
+  * Copyright 2000-2013 JetBrains s.r.o.
+  *
+  * Licensed under the Apache License, Version 2.0 (the "License");
+  * you may not use this file except in compliance with the License.
+  * You may obtain a copy of the License at
+  *
+  * http://www.apache.org/licenses/LICENSE-2.0
+  *
+  * Unless required by applicable law or agreed to in writing, software
+  * distributed under the License is distributed on an "AS IS" BASIS,
+  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
+  * See the License for the specific language governing permissions and
+  * limitations under the License.
+  */
+ package com.intellij.ui;
+
+ import com.sun.jna.Native;
+ import com.sun.jna.NativeLong;
+ import com.sun.jna.platform.unix.X11;
+ import sun.awt.X11.XAtom;
+ import sun.awt.X11.XBaseWindow;
+
+ import java.awt.*;
+
+ /**
+  * Created by IntelliJ IDEA.
+  * Author: Vladimir Kravets
+  * E-Mail: vova.kravets@gmail.com
+  * Date: 4/10/13
+  * Time: 12:19 AM
+  * based on code from VLCJ
+  * http://code.google.com/p/vlcj/source/browse/trunk/vlcj/src/main/java/uk/co/caprica/vlcj/runti
+  */
+
+ public class X11FullscreenHelper {
+   /**
+    * Ask the window manager to make a window full-screen.
+    * <p>
+    * This method sends a low-level event to an X window to request that the
+    * window be made 'real' full-screen - i.e. the window will be sized to fill
+    * the entire screen bounds, and will appear <em>above</em> any window
+    * manager screen furniture such as panels and menus.
+    * <p>
+    * This method should only be called on platforms where X is supported.
+    * <p>
+    * The implementation makes use of the JNA X11 platform binding.
+    *
+    * @param w window to make full-screen
+    * @param fullScreen <code>true</code> to make the window full-screen; <code>false</code> to r
+    * @return <code>true</code> if the message was successfully sent to the window; <code>false</
+    */
+   public static boolean setFullScreenWindow(Window w, boolean fullScreen) {
+     // Use the JNA platform X11 binding
+     X11 x = X11.INSTANCE;
+     X11.Display display = null;
+     try {
+       // Open the display
+       display = x.XOpenDisplay(null);
+       // Send the message
+
+       // Send change property before going to Fullscreen to make sure that WM will know that wir
+       // Workaround for http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=7057287
+       XAtom.get("_NET_WM_STATE").setAtomListProperty((XBaseWindow)w.getPeer(), new XAtom[] {XAtc
+
+       int result = sendClientMessage(
+         display,
+         Native.getWindowID(w),
+         "_NET_WM_STATE",
+         new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
+         x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
+         x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false)
+       );
+       return result != 0;
+     }
+     finally {
+       if(display != null) {
+         // Close the display
+         x.XCloseDisplay(display);
+       }
+     }
+   }
+
+   /**
+    * Helper method to send a client message to an X window.
+    *
+    * @param display display
+    * @param wid native window identifier
+    * @param msg type of message to send
+    * @param data0 message data
+    * @param data1 message data
+    * @return <code>1</code> if the message was successfully sent to the window; <code>0</code> c
+    */
+   private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong dat
+     // Use the JNA platform X11 binding
+     X11 x = X11.INSTANCE;
+     // Create and populate a client-event structure
+     X11.XEvent event = new X11.XEvent();
+     event.type = X11.ClientMessage;
+     // Select the proper union structure for the event type and populate it
+     event.setType(X11.XClientMessageEvent.class);
+     event.xclient.type = X11.ClientMessage;
+     event.xclient.serial = new NativeLong(0L);
+     event.xclient.send_event = 1;
+     event.xclient.message_type = x.XInternAtom(display, msg, false);
+     event.xclient.window = new X11.Window(wid);
+     event.xclient.format = 32;
+     // Select the proper union structure for the event data and populate it
+     event.xclient.data.setType(NativeLong[].class);
+     event.xclient.data.l[0] = data0;
+     event.xclient.data.l[1] = data1;
+     event.xclient.data.l[2] = data2;
+     event.xclient.data.l[3] = new NativeLong(0L);
+     event.xclient.data.l[4] = new NativeLong(0L);
+
+     // Send the event
+     NativeLong mask = new NativeLong(X11.SubstructureRedirectMask | X11.SubstructureNotifyMask);
+     int result = x.XSendEvent(display, x.XDefaultRootWindow(display), 0, mask, event);
+     // Flush, since we're not processing an X event loop
+     x.XFlush(display);
+     // Finally, return the result of sending the event
+     return result;
+   }
+
+   // X window message definitions
+   private static final int _NET_WM_STATE_REMOVE = 0;
+   private static final int _NET_WM_STATE_ADD    = 1;
+
+ }
```

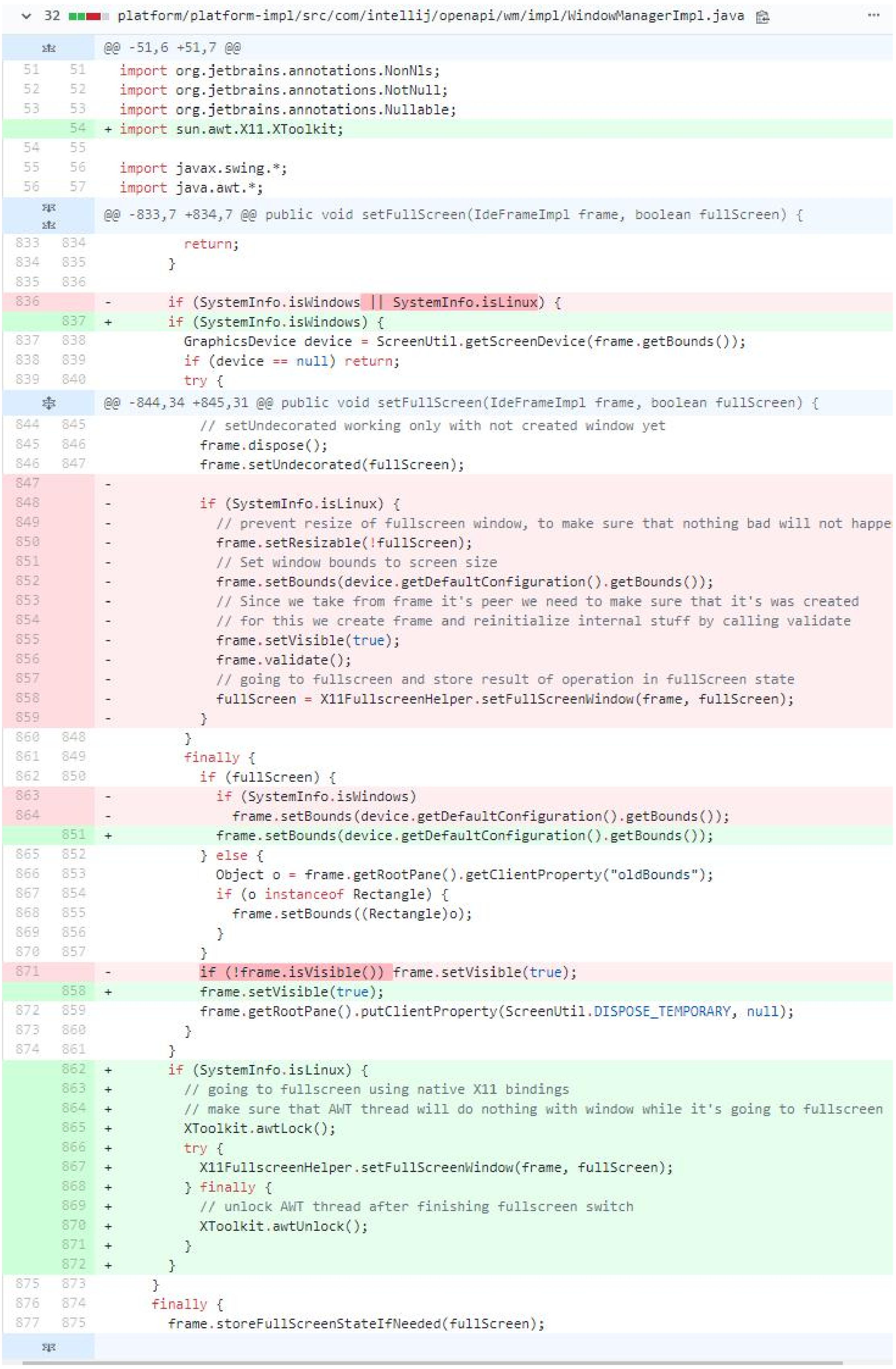
然后应为使用了新的行为实现全屏功能，进行了小重构：
对 Linux 系统单独处理使用新的行为实现制在 linux 系统下的全屏：



```
5 platform/platform-impl/src/com/intellij/openapi/wm/impl/IdeFrameImpl.java
@@ -480,10 +480,13 @@ public boolean isInFullScreen() {
       if (SystemInfo.isMacOSLion) {
         return myFrameDecorator != null && myFrameDecorator.isInFullScreen();
       }
-      if (SystemInfo.isWindows || SystemInfo.isLinux) {
+      if (SystemInfo.isWindows) {
         GraphicsDevice device = ScreenUtil.getScreenDevice(getBounds());
         return (device != null && device.getDefaultConfiguration().getBounds().equals(getBounds())
       }
+      if (SystemInfo.isLinux) {
+        return X11FullscreenHelper.isInFullscreen();
+      }
       return false;
     }
```

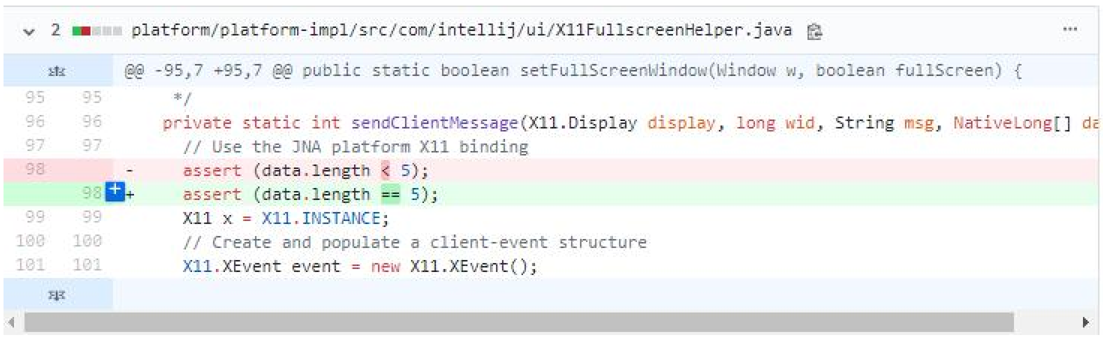


```
35  platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java                ···
@@ -18,8 +18,6 @@
  import com.sun.jna.Native;
  import com.sun.jna.NativeLong;
  import com.sun.jna.platform.unix.X11;
- import sun.awt.X11.XAtom;
- import sun.awt.X11.XBaseWindow;

  import java.awt.*;

@@ -34,6 +32,9 @@
   */

  public class X11FullscreenHelper {
+
+   private static boolean isFullScreenMode = false;
+
    /**
     * Ask the window manager to make a window full-screen.
     * <p>
@@ -59,19 +60,20 @@ public static boolean setFullScreenWindow(Window w, boolean fullScreen) {
        display = x.XOpenDisplay(null);
        // Send the message

-       // Send change property before going to Fullscreen to make sure that WM will know that wir
-       // Workaround for http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=7057287
-       XAtom.get("_NET_WM_STATE").setAtomListProperty((XBaseWindow)w.getPeer(), new XAtom[] {XAtc
-
        int result = sendClientMessage(
          display,
          Native.getWindowID(w),
          "_NET_WM_STATE",
-         new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
-         x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
-         x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false)
+         new NativeLong[]{
+           new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
+           x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
+           x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false),
+           new NativeLong(0L),
+           new NativeLong(0L)
+         }
        );
-       return result != 0;
+       isFullScreenMode = (result != 0) && fullScreen;
+       return (result != 0);
      }
      finally {
        if(display != null) {
@@ -91,8 +93,9 @@ public static boolean setFullScreenWindow(Window w, boolean fullScreen) {
     * @param data1 message data
     * @return <code>1</code> if the message was successfully sent to the window; <code>0</code> c
     */
-   private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong dat
+   private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong[] c
      // Use the JNA platform X11 binding
+     assert (data.length < 5);
      X11 x = X11.INSTANCE;
      // Create and populate a client-event structure
      X11.XEvent event = new X11.XEvent();
@@ -107,11 +110,7 @@ private static int sendClientMessage(X11.Display display, long wid, String ms
      event.xclient.format = 32;
      // Select the proper union structure for the event data and populate it
      event.xclient.data.setType(NativeLong[].class);
-     event.xclient.data.l[0] = data0;
-     event.xclient.data.l[1] = data1;
-     event.xclient.data.l[2] = data2;
-     event.xclient.data.l[3] = new NativeLong(0L);
-     event.xclient.data.l[4] = new NativeLong(0L);
+     System.arraycopy(data, 0, event.xclient.data.l, 0, 5);

      // Send the event
      NativeLong mask = new NativeLong(X11.SubstructureRedirectMask | X11.SubstructureNotifyMask);
@@ -122,6 +121,10 @@ private static int sendClientMessage(X11.Display display, long wid, String ms
      return result;
    }

+   public static boolean isInFullscreen() {
+     return isFullScreenMode;
+   }
+
    // X window message definitions
    private static final int _NET_WM_STATE_REMOVE = 0;
    private static final int _NET_WM_STATE_ADD    = 1;
```
]



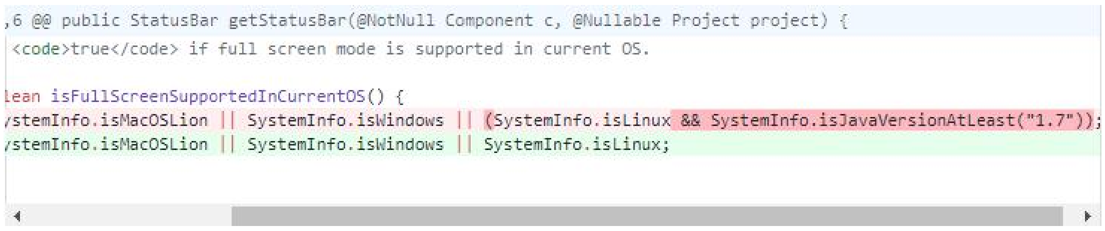
```diff
32  platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -51,6 +51,7 @@
 import org.jetbrains.annotations.NonNls;
 import org.jetbrains.annotations.NotNull;
 import org.jetbrains.annotations.Nullable;
+import sun.awt.X11.XToolkit;
 
 import javax.swing.*;
 import java.awt.*;
@@ -833,7 +834,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
           return;
         }
 
-        if (SystemInfo.isWindows || SystemInfo.isLinux) {
+        if (SystemInfo.isWindows) {
           GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
           if (device == null) return;
           try {
@@ -844,34 +845,31 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
             // setUndecorated working only with not created window yet
             frame.dispose();
             frame.setUndecorated(fullScreen);
-
-            if (SystemInfo.isLinux) {
-              // prevent resize of fullscreen window, to make sure that nothing bad will not happe
-              frame.setResizable(!fullScreen);
-              // Set window bounds to screen size
-              frame.setBounds(device.getDefaultConfiguration().getBounds());
-              // Since we take from frame it's peer we need to make sure that it's was created
-              // for this we create frame and reinitialize internal stuff by calling validate
-              frame.setVisible(true);
-              frame.validate();
-              // going to fullscreen and store result of operation in fullScreen state
-              fullScreen = X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
-            }
           }
           finally {
             if (fullScreen) {
-              if (SystemInfo.isWindows)
-                frame.setBounds(device.getDefaultConfiguration().getBounds());
+              frame.setBounds(device.getDefaultConfiguration().getBounds());
             } else {
               Object o = frame.getRootPane().getClientProperty("oldBounds");
               if (o instanceof Rectangle) {
                 frame.setBounds((Rectangle)o);
               }
             }
-            if (!frame.isVisible()) frame.setVisible(true);
+            frame.setVisible(true);
             frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, null);
           }
         }
+        if (SystemInfo.isLinux) {
+          // going to fullscreen using native X11 bindings
+          // make sure that AWT thread will do nothing with window while it's going to fullscreen
+          XToolkit.awtLock();
+          try {
+            X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
+          } finally {
+            // unlock AWT thread after finishing fullscreen switch
+            XToolkit.awtUnlock();
+          }
+        }
       }
       finally {
         frame.storeFullScreenStateIfNeeded(fullScreen);
```

重构之后又修复了一个关于 assert 的小 bug



```
2 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java
@@ -95,7 +95,7 @@ public static boolean setFullScreenWindow(Window w, boolean fullScreen) {
       */
     private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong[] da
       // Use the JNA platform X11 binding
-      assert (data.length < 5);
+      assert (data.length == 5);
       X11 x = X11.INSTANCE;
       // Create and populate a client-event structure
       X11.XEvent event = new X11.XEvent();
```

之后作者表示在各个桌面管理引擎上测试都通过了，并且由于最后的小重构使得全屏功能在java1.6 上也能使用，所以修改了最初的改动将 java 版本判断去掉了



```
,6 @@ public StatusBar getStatusBar(@NotNull Component c, @Nullable Project project) {
 <code>true</code> if full screen mode is supported in current OS.

lean isFullScreenSupportedInCurrentOS() {
/stemInfo.isMacOSLion || SystemInfo.isWindows || (SystemInfo.isLinux && SystemInfo.isJavaVersionAtLeast("1.7"));
/stemInfo.isMacOSLion || SystemInfo.isWindows || SystemInfo.isLinux;
```

作者文本：

Finally with the last commit, should work with all WMs which support _NET_WM_STATE_FULLSCREEN.

By now check successfully with:

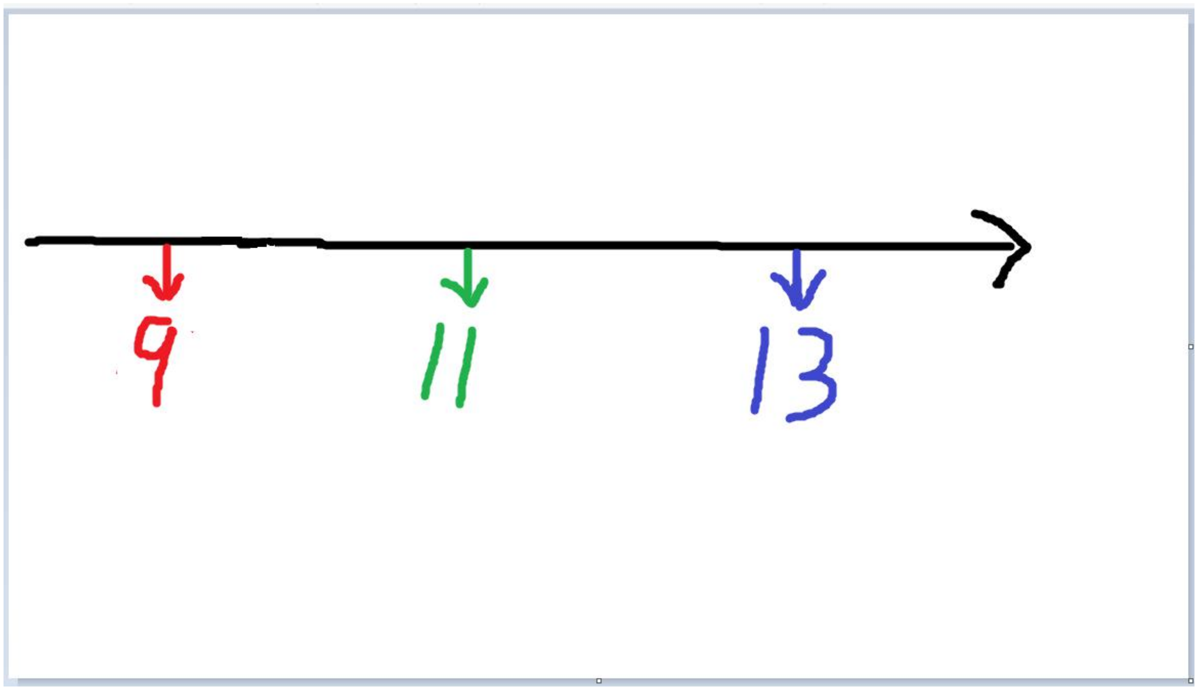
Ubuntu Unity
GNOME3 Shell
SawFish
Enjoy!

Working everywhere. Update status of where its was checked:

Unity
Unity2D
Gnome Classic (Compiz)
Gnome Classic (metacity)
Gnome Shell (Gnome3)
KDE
SawFish
XFCE/XFW
Please review and merge with master

Last behavior of going to Fullscreen mode also support 1.6, 1.7 and 1.8 JRE. Thus checking runtime JRE was removed.

需求变更时间线



9：4 月 9 日修改，给出了基本的判断代码，并对修改代码进行了一些修改，以完成工程开发的需要
11:4 月 11 日发现 Gnome Classic 上修改不工作的问题，修改了在 Gnome Classic 上修改不工作的问题，并相应地重构了代码
13:4 月 13 日修改，完成了在各个平台上的全屏测试，并且得益于 4 月 11 日的修改，将代码试用范围推广至 java1.6