Platform independent key events processing in Java ME

This code example provides a platform independent key mapper for Java ME apps that works across devices from a number of manufacturers.

Problem description

While developing games for multiple J2ME/MIDP enabled mobile devices developers often face to the problem connecting to the identification of pressed keys. This happens for the reason that different mobile phone vendors use different key-codes for the same keys.

Some keys can de defined with `javax.microedition.lcdui.Canvas.getGameAction(int)` method. But some could not, for example: erase key, back-key on SonyEricsson etc. This problem is redoubled in case JAR-file of the game intend for running on mobile phones of different vendors.

Solution

I suggest following solution:

- Create separate class KeyCodeAdapter.
- Define set of internal constants in this class. This constants will be used within game/application.
- Define sets of constants for each mobile vendor, which we want to support in the game/application.

Now we have to determine vendor of mobile phone where application launched. For this we can use following code:

```
    private String getPlatform() {
        // detecting NOKIA or SonyEricsson
        try {
            final String currentPlatform = System.getProperty("microedition.platform");
            if (currentPlatform.indexOf("Nokia") != -1) {
                return PLATFORM_NOKIA;
            } else if (currentPlatform.indexOf("SonyEricsson") != -1) {
                return PLATFORM_SONY_ERICSSON;
            }
        } catch (Throwable ex) {
        }
        // detecting SAMSUNG
        try {
            Class.forName("com.samsung.util.Vibration");
            return PLATFORM_SAMSUNG;
        } catch (Throwable ex) {
        }
        // detecting MOTOROLA
        try {
            Class.forName("com.motorola.multimedia.Vibrator");
            return PLATFORM_MOTOROLA;
        } catch (Throwable ex) {
            try {
                Class.forName("com.motorola.graphics.j3d.Effect3D");
                return PLATFORM_MOTOROLA;
            } catch (Throwable ex2) {
                try {
                    Class.forName("com.motorola.multimedia.Lighting");
                    return PLATFORM_MOTOROLA;
                } catch (Throwable ex3) {
                    try {
                        Class.forName("com.motorola.multimedia.FunLight");
                        return PLATFORM_MOTOROLA;
                    } catch (Throwable ex4) {
                    }
                }
            }
        }
        try {
            if (adaptorCanvas.getKeyName(SOFT_KEY_LEFT_MOTOROLA).toUpperCase().indexOf(SOFT_WORD) > -1) {
                return PLATFORM_MOTOROLA;
            }
        } catch (Throwable e) {
            try {
                if (adaptorCanvas.getKeyName(SOFT_KEY_LEFT_MOTOROLA1).toUpperCase().indexOf(SOFT_WORD) > -1) {
                    return PLATFORM_MOTOROLA;
                }
            } catch (Throwable e1) {
                try {
                    if (adaptorCanvas.getKeyName(SOFT_KEY_LEFT_MOTOROLA2).toUpperCase().indexOf(SOFT_WORD) > -1) {
                        return PLATFORM_MOTOROLA;
                    }
                } catch (Throwable e2) {
                }
            }
        }
        // detecting SIEMENS
        try {
            Class.forName("com.siemens.mp.io.File");
            return PLATFORM_SIEMENS;
        } catch (Throwable ex) {
        }
        // detecting LG
        try {
            Class.forName("mmpp.media.MediaPlayer");
            return PLATFORM_LG;
        } catch (Throwable ex) {
            try {
                Class.forName("mmpp.phone.Phone");
                return PLATFORM_LG;
            } catch (Throwable ex1) {
                try {
```


(C) Copyright Nokia 2013. All rights reserved.

```
                    Class.forName("mmpp.lang.MathFP");
                    return PLATFORM_LG;
                } catch (Throwable ex2) {
                    try {
                        Class.forName("mmpp.media.BackLight");
                        return PLATFORM_LG;
                    } catch (Throwable ex3) {
                    }
                }
            }
        }
        return PLATFORM_NOT_DEFINED;
    }
```

Method above return one of the constants which contains platform name. We can call this method in a constructor of our class KeyCodeAdapter and make this class as singletone. In this way we can not to care of calling getPlatform() method manually, because it will be called automatically after any usage of our class.

Usage example

After actions described above it is enough to transform provided by Canvas native key-codes to our internal. This can be made by simple calling the method adoptKeyCode(), like following:

```
    protected void keyPressed(int keyCode) {
        int internalKeyCode = KeyCodeAdapter.getInstance().adoptKeyCode(keyCode);
        switch (keyCode) {
            case KeyCodeAdapter.SOFT_KEY_LEFT:
                // some processing
                break;
            case KeyCodeAdapter.BACK_KEY:
                // some processing
                break;
            default:
        }
        ...
    }
```

Summary

Full source code you can download here: File:KeyCodeAdapter.zip.

This class can be used for determining vendor's platform of mobile device where application/game is launched by calling method getPlatform.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.