No more portings-designing scalable games

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee (09 Aug 2012)
The article suggests a pattern for allowing you to determine at runtime what code to run, allowing a single JAR to support multiple device configurations. However it is not very clearly described. If this has value it needs to be described better. Ideally with links to other documents on this pattern, and other patterns which can be used to achieve similar results. Title needs to be better too. Something similar to this is covered in Creating a Map Type Selector for the Maps API for Java ME does something similar by detecting the presence of a particular class needed.

Currently many mobile game developers choose the strategy of creating multiple JAR-files for each mobile platform.

But I want to suggest alternative method: one JAR-file for different J2ME platforms. In "different platforms" I mean mobile phones with different screen resolution, heap size, available API's etc.

Tips for designing platform independent applications

==Do not import classes of additional API's==

Instead of this use `Class.forName()` method.

For example:

```
        try {
            Class.forName("com.nokia.mid.ui.FullCanvas");
            return PLATFORM_NOKIA;
        } catch (Throwable ex) {
        }
```

If you use imports your game/application may not install properly on all target devices.

Resolution independent UI and game elements

Game designers have to design screen resolution independent User Interface of the game. This means that coordinates of UI and game elements have not to be hardcoded. Instead of this you have to determine positions and sizes of UI and game elements depending on the current mobile phone's screen resolution.

To make all inscriptions and texts normally readable on target devices use **system fonts**. But pay attention that UI should specialize to the actual font height and width.

Do not use specific platform features

For example:

- Avoid using vibration from Nokia UI API classes, because it doesn't work on Nokia Symbian devices.
- On Samsung devices it is necessary to invoke `System.gc()` method to clear heap memory of unneeded images, but you shouldn't invoke it on other devices to avoid lags.

Use platform independent controls

How to use it is described in the article: Platform independent key events processing in Java ME

Test your game/application on each target platform

To agree that your game/application works properly on all target devices it is recommended to test game/application on a real devices or use Forum.Nokia Program Remote Device Access.

Benefits of platform independent games/applications

- Reducing cost and estimation of a game/application development;
- Simplification of a product support for development and marketing teams because of one JAR-file;
- Same game features for all platforms;
- Simplification of a distribution;
- Product evolution and improvement simplification;

Shortcomings

- Insignificant increasing of JAR-file size, because functionality for all platforms is included to only build;
- Developer is limited with the most "weak" device from a supported range;

Summary

As we can see, designing of platform independent games/applications is real alternative for multiple JAR-files strategy which can appreciably reduce estimation of development and as consequence to reduce development cost.