Chapter 14

The Proper Java ME Mindset

So far in the book, we have primarily discussed the technical side of creating a Java ME application. While this is clearly the most important aspect of Java ME application development, there is another aspect that can sometimes be just as important as the technical aspect: the mindset with which you approach the development process.

In order to understand why having the proper mindset is important for Java ME development, it’s important to note that every software platform out there, mobile or otherwise, tends to have a specific development methodology associated with it. This is either imposed by the platform’s creators (by providing a “best practices” document and structuring the API accordingly) or tends to grow organically as the platform gains traction and evolves. Furthermore, the development methodology tends to change with time, as the platform’s scope and underlying hardware technology change.

Java ME is somewhat unique in these respects. It was initially conceived to run on extremely limited devices but is now supported on (comparatively) very powerful ones too; and in practice you frequently have to support both categories of hardware at the same time. As both old and new Java ME hardware is in use today, there are no universal “best practices” you can follow, nor is there a universal development methodology; what works on old hardware may not be effective on new hardware and vice versa. There are some general tips you can take into consideration in all situations (the most important of which have been enumerated in Chapter 1), but that’s about it.

Then there is the Java ME API itself, which is designed to be as simple as possible in order to be usable on as many devices as possible. This means that, in order to do more complex stuff, you frequently have to be creative and implement by yourself many features that you would take for granted on other platforms, from simple utility methods (like string matching) to complex functionality such as serialization. Deciding which of these features you need to implement in order to meet your goals (while being mindful of the project’s restrictions in terms of resources and target hardware), and how to implement them, is often an art in itself—and the decisions made in this area affect the development process as a whole.
Finally, you also have to consider the type of application you are writing, as this often dictates the way you divide your available resources between the UI, flexibility, and functionality. Sometimes the UI is more important than the actual functionality, while other times it’s the other way around, or the flexibility of the application (its ability to run in many different environments) is its most important trait. Making the right decision here is crucial for the success of your application.

For these reasons, having a proper mindset—a core set of principles that dictate how you approach the development of any given application—is crucial for the success of any Java ME project. These principles help you understand what Java ME as a platform can and cannot do in a given scenario, and they can help you determine what you should and should not do in a given situation.

In this chapter, we will take a look at what the proper Java ME development mindset is, and we will try to demonstrate it with a few real-life examples.

NOTE: Of course, the principles described in this chapter can be applied to other platforms as well. However, they apply to other platforms to a much lesser degree than they do to Java ME; Java ME stands to gain the most from them simply because of its more constrained and less fault-tolerant nature.

Java ME Is Only As Powerful As the Devices It Runs On

It is surprising how many people tend to judge the Java ME platform separately from the underlying hardware. Sure, it feels intuitive to judge a platform based solely on its specifications, but with Java ME this is simply wrong. The reason for this is that, while Java ME itself isn’t exceptionally gifted in terms of capabilities, it does have provisions for improving and extending said capabilities—and these provisions can make a world of difference.

For example, Java ME’s default graphical capabilities are extremely limited by today’s standards. However, as we have seen in the previous chapter, you can extend these capabilities significantly and bring Java ME to the same level as more modern platforms. The only limiting factor in this endeavor is the underlying hardware’s processing power. In other words, most of the time it’s not a question of whether it can be done with Java ME (it usually can be done); it’s a question of whether it can be done on the hardware at hand. The same applies to many other features or capabilities that Java ME lacks or that are underdeveloped.

Also, while the device’s hardware is extremely important, equally important are the optional capabilities that said device supports. This includes both optional JSRs and vendor-specific extensions, both of which can make a world of difference in deciding the feasibility of a project. Some JSRs and extensions are critical for a given project; for example, the File Connection API support is critical if you plan on writing a file manager