A Watch - from an OOP Perspective

Adrian Kosmaczewski

2008-07-13

A watch might be one of the most common types of objects, but it remains also one of the earliest pieces of human craftmanship to show an extreme level of complexity, all contained in a small amount of space. Since the late 1700s, artisan watchmakers in Switzerland and elsewhere have shown their pride and skills creating watches called “Grande Complications”, containing thousands of individual parts and performing incredible functions:

> The most complicated watch ever made, known in watch enthusiasts' circles as “The Ultimate Watch,” is Patek Philippe's “Calibre 89.
>
> The incredibly precise operation of 1728 parts in this really ultimate masterpiece of watchmaking allows to perform no less than 33 (thirty-three!) complicated functions, among them a correction for the 400-year-rule, an Easter date indication, a star chart, a tourbillon, a perpetual calendar, a sidereal time indication, and, and, and …
>
> This watch was sold in 1989 for the nice round sum of about four million Swiss francs.

(Ozdoba, 2005)

(Source: CNN.com, 2005)

More information about the “Calibre 89” can be found here and in the Patek Philippe Museum website.

However, the same watchmakers that made these fine pieces were also aware of the basic information that their creations were to provide: time. As such, their watches remained extremely easy to use, and they set up the basic standard for analog watches, in such timeless designs that the latest Swatch models show the same basic layout and functionality.

The underlying concept is the very same used in today's object-oriented abstraction and encapsulation. Even Apple uses the idea of the watch to show this characteristic:

All programming languages provide devices that help express abstractions. In essence, these devices are ways of grouping implementation details, hiding them, and giving them, at least to some extent, a common interfaceâ€"much as a mechanical object separates its interface from its implementation, as illustrated in Figure 2-1.

(Source: Apple Developer Connection, 2006)

In this article I will provide my view about how different OOP concepts apply to a real-life object such as a watch, in all its forms.

Concepts

Object

Every existing watch could be thought of a single instance of a more generic class "Watch". This is possible since all watches share a common set of characteristics, that is, at least, the capacity to display the current time. Of course their external appearance and their whole set of characteristics may differ, but in this common aspect, they are all the same. This makes the "Watch" class a simple but extensible one.

Attributes

Every watch has a distinct set of attributes, for example:

- Brand
- Type of wristband (metallic, leather, plastic)
- Waterproofness
- Current time
- Size
- Cadran color
- Analogic or Digital
- Number of hands
- Battery level

Behavior

Watches have a basic behavior; they increment their "current time" attribute, by one second every second. This way, they manage to update themselves automatically, and to provide users with the right time all the time. Watches that provide different functionalities may also have different behaviors (I had long ago a Casio watch which provided barometric pressure and temperature, both constantly updated and very handy when doing outdoor activities).

Class

A “Watch” class would provide common characteristics to all watch instances, the most basic being that of providing time information and being able to be held by a person (his owner).

Inheritance

There are lots of different types of watches, but some common subsets can be easily seen:

- Swiss vs. Japanese
- Digital vs. Analog watches
- “Grande complication” vs. simpler watches
- Battery vs. wrist kynetics- powered
- Plastic vs. metal watches
- Quartz vs. mechanical

These subsets can be used to model the right class inheritance, the one that makes more sense in an application (since there is usually not a single answer). C++ provides also multiple inheritance, which is not the case of many languages; in this case, every instance could inherit from several base classes (Japanese + digital + simpler + battery + plastic + quartz = Casio watch).

At the same time, a “Watch” could be seen as a subclass of “Clock”, which also provides time information but is usually not easy to hold in the palm of your hand (the Big Ben is a clock, but not a watch).

Abstraction

Users are completely isolated from the internal mechanisms of their watch; they usually don’t know (nor they need to know) how their watches work; they just care about the current time displayed.

Modeling

When creating a new kind of watch, a designer might find inspiration on existing watches or from new functionalities that might be useful to the end user. This in turn “reshapes” an implicit Watch class, adding new subclasses, or properties to existing classes. This is something that Casio made extensively in the 80s and 90s, providing watches with extended functionalities, rather different from what was offered at the time (temperature, barometric pression, speed, even television or radio).

Messages

Whenever the user sets up the right time in a watch, or uses one of its functionalities (for example to get the current date, the temperature), she has to interact somehow with the watch; usually this is done using a set of controls (knobs or keys) located around or over the watch, and each time that the user presses or turns one of those controls, one could think of a message sent to the watch. Of course a correct protocol is needed (that is, turn or press twice to set the time to 2 o'clock) and some operations are forbidden, or even better, impossible (like setting the hour to 25, or the minutes to 345).

Encapsulation

This concept is closely related to that of "Abstraction"; users can use and interact with rather complex structures, using extremely simple commands, like buttons and knobs. This encapsulation guarantees a correct mechanism (it is difficult for users to screw up their watches), satisfies warranty requirements (since only qualified people can deal with the internal structure of the watches), and also simplifies the use of the watch, making it easy to use and providing value to the user.

Interface

Users do not need to know whether Patek Philippe watches are more complicated than Swatches, simply because their basic functionality is the same, and the difference has more to do with aesthetics (and price…) than anything else. Both provide the time in analogic form, with similar knobs and maybe even similar advanced functionalities (date, chronograph, etc).

Information hiding

The watch displays only the right amount of information to the user; the rest is kept internally, and is used by the internal mechanisms without the user even knowing about them.

Data members

Inside the mechanisms of the watch, implicit (in the case of analog watches) or explicit (in the case of digital watches) bits of information can be found; these could be either private (angles of the hands, resort tensions, controller values) or public (current time, battery level).

Member functions

Every distinct functionality or service of a watch could be thought of like a "member function" of the class Watch:

- Display current time
- Set time to new value
- Display current date
- Start chronograph

- Buzz alarm at the right time

Internally, the watch may have other “private” member functions, used to perform internal duties.

References

Apple Developer Connection, “Interface and Implementation”, [Internet] http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/Articles/chapter_3_section_2.ht
(Accessed October 1st, 2006)

CNN.com, “Patek Philippe – The ultimate watch”, Monday, October 31, 2005, [Internet] http://edition.cnn.com/2005/WORLD/europe/10/27/ultimate.watch/index.html (Accessed October 1st, 2006)

Ozdoba, Christoph, “Pocket Watches - Glossary”, August 13, 2005, [Internet] http://www.ozdoba.net/swisswatch/pocket_gloss.html#comp (Accessed October 1st, 2006)

Patek Philippe Museum, [Internet] http://www.patekmuseum.com/ (Accessed October 1st, 2006)

Swatch website, [Internet] http://swatch.com/index_flash.php (Accessed October 1st, 2006)

Webb, Ken, “Digital Watch - Sample Xholon App”, [Internet] http://www.primordion.com/Xholon/samples/wa
(Accessed October 1st, 2006)