Subject: Re: subclassing LineEdit is ugly
Posted by mirek on Mon, 10 Apr 2006 08:08:27 GMT
View Forum Message <> Reply to Message

Quote:
Let me put an example. Somebody wants a LineEdit in which Shift-Tab unindents only if none of the selected lines are already fully unindented. It is clearly a subclassing situation: you are replacing only part of the behaviour. But AlignChar is non-virtual for whatever reason. So you need to do all these steps:
1) create a new method instead of AlignChar
2) override the method Key, copy-paste long code from LineEdit
3) call your new method from Key, instead of AlignChar
4) pray that no further subclasses, or methods of your subclass would accidentally call AlignChar instead of your new method
5) every time a new version of library comes out try to sync the changes to the copy-pasted Key method

Well, let us discuss it:

1 - you have to add method anyway
2&3 - nope. Just override Key, react to Shift-Tab by calling your new method, for other keys call original LineEdit::Key.

```
bool MyLineEdit::Key(dword key, int count)
{
  if(key == K_SHIFT|K_TAB) {
     MyAlignChar();
     return true;
  }
  return LineEdit::Key(key, count);
}
```

4 - or pray that no other code does expect original AlignChar behaviour... See, by not makeing some methods virtual, you enforce stronger contract - you guarantee single behaviour.

5 - nope because of 2.

Quote:
My reasoning for this is that when you are developping an application yourself and need a subclass it is very easy for you to just make the needed method virtual in the base class, or just insert a branch in the library code itself. But for the clients of the library we are stuck with the amount of flexibility which is readily provided by the library. Imaging working in an environment where you can not change the library, but required to provide slighly different behaviour in some classes. This different working method places different requirements on the library, which may not be realized by you while working on one of your own applications.

OK, I will think about it. However, all my experiences favor "black box" approach, with as narrow interfaces as possible. It means, if problem can be solved by composition rather than subclassing, I favor composition.

In this particular case (using LineEdit for terminal, I have no further description, so I may be wrong) I am pretty sure that things can be solved pretty easy without adding virtual methods. That said, if you want to push me into another direction, please provide more detailed description (adding characters at the end of console window only is too trivial to justice adding 10 virtual methods

Once again, it is not about speed concerns, but interface definition. I simply believe that for majority of methods it is much bettter when they are "final".

BTW, the real problem there is that AlignChar is protected. It should in fact be private...

Mirek
