On Making Code More Readable

By Luboš Luňák
Software Developer at Collabora Productivity
A Quiz !
#1 What is the purpose of this class?

struct FormulaGroupContext
{
    typedef AlignedAllocator<double, 256> DoubleAllocType;
    ...
    ColArray* getCachedColArray( SCTAB nTab, SCCOL nCol,
        size_t nSize );
    ColArray* setCachedColArray(
        SCTAB nTab, SCCOL nCol, NumArrayType* pNumArray,
        StrArrayType* pStrArray );
    void ensureStrArray( ColArray& rColArray, size_t
        nArrayLen );
    void ensureNumArray( ColArray& rColArray, size_t
        nArrayLen );
    ...
}
#2 Why are the classes added?

commit xxxxx : sc: replace ScCaptionPtr with std::shared_ptr, tdf#117997, tdf#117228

+class ScUndoDelSdrCaptionObj: public ScUndoSdrCaptionObj
  + [copy&paste from SdrUndoRemoveObj]
...
+class ScUndoNewSdrCaptionObj: public ScUndoSdrCaptionObj
  + [copy&paste from SdrUndoInsertObj]
...

Comments added by the commit: 0
#3 What does this class name mean?

SwSpzFrmFmts
Code Readability Matters
STL containers

if( find( mylist.begin(), mylist.end(), 10 ) != mylist.end )
{
    double value = mystack.top();
    mystack.pop();
    return value;
}
Qt containers

if( mylist.contains( 10 ))
    return mystack.pop();
 STL strings

bool findStringIC(const std::string & haystack, const std::string & needle)
{
    auto it = std::search( 
        haystack.begin(), haystack.end(), 
        needle.begin(), needle.end(), 
        [](char ch1, char ch2) { return std::toupper(ch1) == 
            std::toupper(ch2); } 
    );
    return (it != haystack.end() );
}

(https://stackoverflow.com/questions/3152241/case-insensitive-stdstring-find)
Qt strings

haystack.contains( needle, Qt::CaseInsensitive );
Code is read more often than written

Code is read often

- Reviewing, fixing bugs, figuring out how it works
- Code is read even when (re)writing it

Optimize primarily for reading, not for writing

- If that’s where more time is spent, then that is what should be optimized for
- And that is what should give better returns
- Often requires initial investment (time, effort)
- Poor code readability can cost a lot of time wasted
- Technical debt may accumulate too much
Existing LO improvements
LO strings

Before:

```cpp
rtl::OUString aFileName = rtl::OUStringBuffer()
    .appendAscii(RTL_CONSTASCII_STRINGPARAM("charts/chart"))
    .append(nCount)
    .appendAscii(RTL_CONSTASCII_STRINGPARAM(".xml"))
    .makeStringAndClear();
```

Now:

```cpp
OUString aFileName = "charts/chart" + OUString::number(nCount) + ".xml";
```
SvStream binary read/write

Before:

rStream << value;

Now:

rStream.writeDouble( value );
Tinderbox/Jenkins

Before:
- Repository builds repeatedly broken
- Windows builds occasionally broken for a long time
- Problems with (bi)bisecting

Now:
- Code always[*] builds on all platforms
- [*] - almost :)

Unit tests

Before:

- Unit tests that almost nobody knew how to use
- And almost nobody ran them
- Changes breaking other changes

Now:

- Many Writer, Calc, etc. unit tests, export/import, features, ...
How the code improves
Make it easy to do things the right way

- LO strings are easier to read and write than OOo strings
- Clang plugins check quality and make it easier to convert code
- Gerrit/Jenkins makes it easy to check if code compiles, on all platforms
- Existing unit tests and documentation makes it (reasonably easy) to write more unit tests

These are usually investments that return over time.
How can we improve more

Mindset matters.

- It should not be ok to do things the bad way.
- Commit logs should not only say what but also why.
- Classes should have at least a brief summary of its purpose.
- Writing down a comment can help already while writing the code.
- ...?

Tools can help with making things in a better way.

- Gerrit check for undocumented classes/code?
- ...?
Thank you!