Professional Issues II
Unit 2: code commenting
The idea in SE context

Markus Roggenbach

Janary 2020

On the topic of bugs

Deutsche Bahn rejects 25 Bombardier trains over 'manufacturing defects'

The main problems were with the on-board computers' operating system, which was said to "crash regularly".

https:

//www.thelocal.de/20200128/deutsche-bahn-rejects-25-bombardier-trains-over-manufacturing-defects

My offer concerning off-task media use

Have the first 2 rows in the room device fee zone.

Recap

- Software Engineering – systematic development of software products.
- Software maintenance cost dominates development cost.
- In maintenance, 50% of time spent in the process of understanding the code.

You will learn

How to comment on software such that it eases understanding of the code.

In particular, we look into

- Documentary comments – basic information on a file
- Structural comments – functionality of single methods

A. Commenting Code: documenting & class/method structure

Why commenting?

```
#!/usr/bin/perl -s
## Ian Goldberg <ian@cypherpunks.ca>, 19980817
$f=$d?-1:1;4D=pack(C',33..86);$p=shift;
$p=~y/a-z/A-Z/;$U='$D=~s/.*)U$/U$1/;
$D=~s/U(.)/$1U/;';($V=$U)=~s/U/V/g;
$p=~s/[A_Z]/$k=ord($&)-64,&e/eg;$k=0;
while(<>){y/a-z/A-Z/;y/A-Z//dc;$o.=$_}$o='X'
while length($o)%5&&!$d;
$o=~s/./chr(($f*&e+ord($&)-13)%26+65/eg;
$o=~s/X*$//if $d;$o=~~s/.{5}/$& /g;
print"$o/n";sub v{$v=ord(substr($D,$_[0]))-32;
$v>53?53:$v}
sub w{$D=~s/(.{$_[0]})(.*)(.)/$2$1$3/}
sub e{eval"$U$V$V";$D=~s/(.*)([UV].*[UV])(.*]/$3$2$1/;
&w(&v(53));$k?(&w($k)):($c=&v(&v(0)),$c>52?&e:$c)}
```

Some Guidelines

- Robert S Laramee:

  A Source Code Comment Standard

  http://www.cs.swan.ac.uk/~csbob/teaching/laramee07commentConvention.pdf

- *13 Tips to Comment Your Code*

  http://www.devtopics.com/13-tips-to-comment-your-code/

- Bernhard Spuida:

  The fine Art of Commenting

  http://www.icsharpcode.net/TechNotes/Commenting20020413.pdf

- Stackoverflow:

  Are there standard formats for comments within code?

  https://stackoverflow.com/questions/779025/are-there-standard-formats-for-comments-within-code

Category 1 of comments: Documentary comments

1. File-name
2. Version number/build number
3. Creation date
4. Last modification date
5. Author's name
6. Copyright notice
7. Purpose of the program
8. Version history

Documentary comments

- At the start of any program file.
- Provides basic information on the file.

Category 2 of comments: structure description

Typical example: Java class `StdIn`

Method Summary

Methods

Modifier and Type	Method and Description
`static boolean`	**`hasNextChar()`** Returns true if standard input has more inputy (including whitespace).
`static boolean`	**`hasNextLine()`** Returns true if standard input has a next line.
`static boolean`	**`isEmpty()`** Returns true if standard input is empty (except possibly for whitespace).

Structure description

- Provide an overview on what functionality is available in the form of methods.
- Concise descriptions explain how to use the functionality.

MR's Commenting Standard – Part 1: Method declarations

Standard notation in CS

<. . . > – means: the part between "<" and ">" is to be replaced by a string that fits the description given by ". . ."

Example: <first name><family name>

can be expanded to, e.g.,

- Markus Roggenbach
- Randall Gaya
- Jean Razafindrakoto

Method declaration, abstractly:

```
<modificators> <result-type> <method-name>
  ( <type 1> <formal-parameter 1> ,
    ...
    <type n> <formal-parameter n>  ) {
 ...
}
```

Example:

```
    public static int mortgage (int age, int salary) {
        ....
    }
```

Required commenting

- Describe in "natural language" what the method does.
- Formula how the result is computed out of the parameters (if possible).
- Explain the parameters.
- State if there are side-effects.
- State if the method is referentially transparent.

Example

```
/* computes the maximal mortgage for a customer   */
/* based on                                       */
/* int age    - the age (1..100) of the customer  */
/* int salary - the regular annual income         */
/*              in thousand Pounds                */
/* no side-effects                                */
/* referentially transparent                      */


    public static int mortgage (int age, int salary) {
        ....
    }
```

Excursion: Side effect

A method is side-effect free, if it does not change the global state of a program.

```
int x;

public static int Hugo () {
  x = 25;
  return 17;
}
```

Hugo changes the global variable x, thus it has a side-effect.

Excursion: Referentially transparent

A method is referentially transparent, if its return value (with identical actual parameters) is independent of the program context.

```
int y;

public static int Erna () {
  return y;
}
```

The result of `Erna` is the content of the global variable `y`, thus it is not referentially transparent.

What you have learned in this unit

Definitions

- Documentary comments
- Structural comments
- Referential transparency
- Side effect

You should be able to explain by example

- when is a method referential transparent?
- when does a method have a side effect?