SE302 – ПХ БҮТЭЭЛТ
Лекц 12

Programming Style as Documentation

- Дотоод баримтжуулалт (Internal Documentation)
  - Эх кодын түвшинд
- Кодын түвшиний баримтжуулалтын гол оролцогч нь тайлбар биш
  - Not comment
- Good Programming Style
  - Энгийн, хялбар ойлгохоор
  - Сайн хувьсагчын нэр
  - Сайн функцын нэр
  - Тогтмол ашиглах
  - Удирдлагын урсгал болон ӨБүтэцийн төвөгтэй байдлыг бууруулах

Programming Style as Documentation

Java Example of Poor Documentation Resulting from Bad Programming Style

```
for ( i = 2; i <= num; i++ ) {
meetsCriteria[ i ] = true;
}
for ( i = 2; i <= num / 2; i++ ) {
j = i + i;
while ( j <= num ) {
meetsCriteria[ j ] = false;
j = j + i;
}
}
for ( i = 2; i <= num; i++ ) {
if ( meetsCriteria[ i ] ) {
System.out.println ( i + " meets criteria." );
}
}
```

Programming Style as Documentation

Java Example of Documentation Without Comments (with Good Style)

```
for ( primeCandidate = 2; primeCandidate <= num; primeCandidate++ ) {
   isPrime[ primeCandidate ] = true;
}

for ( int factor = 2; factor < ( num / 2 ); factor++ ) {
   int factorableNumber = factor + factor;
   while ( factorableNumber <= num ) {
      isPrime[ factorableNumber ] = false;
      factorableNumber = factorableNumber + factor;
   }
}

for ( primeCandidate = 2; primeCandidate <= num; primeCandidate++ ) {
   if ( isPrime[ primeCandidate ] ) {
      System.out.println( primeCandidate + " is prime." );
   }
}
```

Comments or Not to Comment

- Тайлбар их хийх нь кодыг замбараагүй бичихэд хүргэдэг
  - Туслахаасаа илүү хор учруулах

Keys to Effective Comments

Java Mystery Routine Number One

```
// write out the sums 1..n for all n from 1 to num
current = 1;
previous = 0;
sum = 1;
for ( int i = 0; i < num; i++ ) {
   System.out.println( "Sum = " + sum );
   sum = current + previous;
   previous = current;
   current = sum;
}
```

Keys to Effective Comments

- Хэр сайн тайлбар вэ?

Java Mystery Routine Number Two

```
// set product to "base"
product = base;

// loop from 2 to "num"
for ( int i = 2; i <= num; i++ ) {
   // multiply "base" by "product"
   product = product * base;
}
System.out.println( "Product = " + product );
```

Keys to Effective Comments

- Хэр сайн тайлбар вэ?

Java Mystery Routine Number Three

```
// compute the square root of Num using the Newton-Raphson approximation
r = num / 2;
while ( abs( r - (num/r) ) > TOLERANCE ) {
    r = 0.5 * ( r + (num/r) );
}
System.out.println( "r = " + r );
```

Keys to Effective Comments

- Ж1: Тайлбар алдаатай
- Ж2: Кодыг нуршиж тайлбарласан. Үр дүн муутай
- Ж3: Тохирсон тайлбарыг өгсөн

- Муу тайлбар нь тайлбаргүй байснаас илүү муу
  - Ж1, Ж2 нь тайлбаргүй байсан нь илүү дээр

Kinds of Comments

- Тайлбаруудыг 6 бүлэгт хувааж үзнэ
- Repeat of the code
  - Кодыг өөр үгээр тайлбарлах
  - Уншигчид илүү зүйл уншихаар өгөхөөс биш нэмэлт мэдээлэл байхгүй
- Explanation of the code
  - Ихэвчлэн нарийн төвөгтэй, эмзэг хэсэгт тайлбарладаг. (Хэрэгцээтэй)
  - Тайлбар бичихийн оронд түүнийг энгийн ойлгожтой болгож сайжруулах нь илүү үр дүнтэй
    - Ойлгомжтой болгоод дүгнэлт, тайлбар оруулах

Kinds of Comments

- Marker in the Code
  - Гүйцэт дуусгаагүй ажлыг тэмдэглэж үлдээх

```
return NULL; // ****** NOT DONE! FIX BEFORE RELEASE!!!
```

    - ********
    - !!!!!!!!!!
    - TBD
  - TO DO - зарим Editor-ууд дэмждэг

Kinds of Comments

- Summary of the Code
  - Кодыг дүгнэж тайлбар хийх
    - Олон мөр кодыг нэг эсвэл хоёр мөр өгүүлбэр болгох
  - Ийм төрлийн тайлбар нь илүү үр дүнтэй
    - Үгчилж тайлбарлахаас илүү
    - Уншигч кодноос илүү хурдан уншиж чадна
  - Ийм төрлийн тайлбар нь зохиогчоос өөр хүн тухайн кодонд өөрчлөлт хийхэд илүү үр өгөөжтэй

Kinds of Comments

- Description of the Code's intent
  - Кодын хэсгийн зорилгыг тайлбарлах
  - Шийдлээс илүү асуудлын түвшинд илүү үр дүнтэй
    - Intent comment

```
-- get current employee information
```

    - Summary comment

```
-- update employeeRecord object
```

- IBM-ын явуулсан 6 сарын сургалтаар
  - Програмистууд зорилгыг ойлгох нь хамгийн хэцүү байсан
    - Зорилго болон дүгнэлт тайлбарын ялгаа нь үргэлж тодорхой биш

Kinds of Comments

- Information That Cannot Possibly Be Expressed by the Code Itself
  - Зарим мэдээллийг кодонд тусгах шаардлагагүй ч эх кодонд агуулах ёстой
    - Зохиогчын эрхийн мэдэгдэл
    - Нууцлалын мэдэгдэл
    - Хувилбарын дугаар
    - Кодын дизайны тухай
    - Архитектурын баримт

Kinds of Comments

- Information That Cannot Possibly Be Expressed by the Code Itself

C++ Example of a Commenting Style That's Easy to Maintain

```
/*****************************************************************************
   class:  GigaTron (GIGATRON.CPP)

   author: Dwight K. Coder
   date:   July 4, 2014

   Routines to control the twenty-first century's code evaluation
   tool. The entry point to these routines is the EvaluateCode()
   routine at the bottom of this file.
*****************************************************************************/
```

Commenting Paragraphs of Code

- Сайн баримтжуулсан програмд
  - 1 эсвэл 2 мөр өгүүлбэр нь кодын параграфуудыг тодорхойлно

Java Example of a Good Comment for a Paragraph of Code

```
// swap the roots
oldRoot = root[0];
root[0] = root[1];
root[1] = oldRoot;
```

Commenting Paragraphs of Code

- Write comments at the level of the code's intent
  - Зорилгын түвшинд нь нөлөөлж чадаагүй

Java Example of an Ineffective Comment

```
/* check each character in "inputString" until a dollar sign
is found or all characters have been checked
*/
done = false;
maxLen = inputString.length();
i = 0;
while ( !done && ( i < maxLen ) ) {
   if ( inputString[ i ] == '$' ) {
      done = true;
   }
   else {
      i++;
   }
}
```

Commenting Paragraphs of Code

- Focus your documentation efforts on the code itself

Java Example of a Good Comment and Good Code

```
// find the command-word terminator
foundTheTerminator = false;
commandStringLength = inputString.length();
testCharPosition = 0;
while ( !foundTheTerminator && ( testCharPosition < commandStringLength ) ) {
   if ( inputString[ testCharPosition ] == COMMAND_WORD_TERMINATOR ) {
      foundTheTerminator = true;
      terminatorPosition = testCharPosition;
   }
   else {
      testCharPosition = testCharPosition + 1;
   }
}
```

Commenting Paragraphs of Code

- Хэрхэн гэдгээс илүү яагаад дээр тайлбарыг төвлөрүүлэх

Java Example of a Comment That Focuses on *How*

```
// if account flag is zero
if ( accountFlag == 0 ) ...
```

Java Example of a Comment That Focuses on *Why*

```
// if establishing a new account
if ( accountFlag == 0 ) ...
```

Java Example of Using a "Section Heading" Comment

```
// establish a new account
if ( accountType == AccountType.NewAccount ) {
   ...
}
```

Commenting Paragraphs of Code

- Commenting Data Declarations

Visual Basic Example of Nicely Documented Variable Declarations

```
Dim cursorX As Integer  ' horizontal cursor position; ranges from 1..MaxCols
Dim cursorY As Integer  ' vertical cursor position; ranges from 1..MaxRows

Dim antennaLength As Long      ' length of antenna in meters; range is >= 2
Dim signalStrength As Integer  ' strength of signal in kilowatts; range is >= 1

Dim characterCode As Integer      ' ASCII character code; ranges from 0..255
Dim characterAttribute As Integer ' 0=Plain; 1=Italic; 2=Bold; 3=BoldItalic
Dim characterSize As Integer      ' size of character in points; ranges from 4..127
```

Commenting Paragraphs of Code

C++ Example of Commenting the Purpose of a Control Structure

```
// copy input field up to comma
while ( ( *inputString != ',' ) && ( *inputString != END_OF_STRING ) ) {
   *field = *inputString;
   field++;
   inputString++;
} // while -- copy input field

*field = END_OF_STRING;

if ( *inputString != END_OF_STRING ) {
   // read past comma and subsequent blanks to get to the next input field
   inputString++;
   while ( ( *inputString == ' ' ) && ( *inputString != END_OF_STRING ) ) {
      inputString++;
   }
} // if -- at end of string
```

Commenting Paragraphs of Code

Visual Basic Example of a Monolithic, Kitchen-Sink Routine Prolog

```
'*********************************************************************
' Name: CopyString
'
' Purpose:      This routine copies a string from the source
'               string (source) to the target string (target).
'
' Algorithm:    It gets the length of "source" and then copies each
'               character, one at a time, into "target". It uses
'               the loop index as an array index into both "source"
'               and "target" and increments the loop/array index
'               after each character is copied.
'
' Inputs:       input    The string to be copied
'
' Outputs:      output   The string to receive the copy of "input"
'
' Interface Assumptions: None
'
' Modification History: None
'
' Author:       Dwight K. Coder
' Date Created: 10/1/04
' Phone:        (555) 222-2255
```