MEIC - Engenharia de Software – 2011/12

Instructions for Program 1

**Read this document completely and clarify any question with your instructor before starting the work!**

# Goals

Become used to develop software according to a defined process; understand what a process definition looks like; gather and analyze basic data about the work and the work products for understanding and improving your performance.

# Requirements

Using the software development process SDP1 described next, write a program (LOCCounter) in Java to count the number of lines of code (LOC) of a set of Java source files, ignoring blank lines and comments. The names of the source files are passed as command line parameters.

The program should print the size (LOC) of each file (ignoring blank lines and comments) as well as the total size. It should also support a “-p” option to print the source code prepended with line numbers, after ignoring blank lines and comments.

Thoroughly test the program. At a minimum, run the following test cases.

|  |  |  |
| --- | --- | --- |
| # | Input (command line and source file) | Expected output |
| 1 | java LOCCounter -p Class1.java  /\*\*  \* Simple program.  \*/  public class Class1 {    public static void main() {  // print  System.out.println(x);  }  } | Class1.java: 5 LOC  1: public class Class1 {  2: public static void main() {  3: System.out.println(x);  4: }  5: }  Total: 5 LOC |
| 2 | java LOCCounter -p File2.java  /\*/ false end of comment  \*/  x = "/\*"; // false start of comment  y = "\""; /\* false end of string \*/  z = '"'; /\* false start of string \*/  w = /\* line merge comment  \*/ 1; | File2.java: 4 LOC  1: x = "/\*";  2: y = "\"";  3: z = '"';  4: w = 1;  Total: 4 LOC |
| 3 | java LOCCounter Class1.java File2.java | Class1.java: 5 LOC  File2.java: 4 LOC  Total: 9 LOC |

To promote reuse and extensibility, it is recommended that you separate the main program building blocks in different methods or classes. In the design phase, it may be helpful to draw a UML state machine diagram describing comment processing.

In the end of this assignment, you’ll have to use your program to measure the size of itself.

# SDP1 Process scripts

Before starting program 1, review the top-level process script below to ensure that you understand the “big picture” before you begin. Also, ensure that you have all of the required inputs before you begin the planning phase.

## SDP1 Process Script

|  |  |
| --- | --- |
| Purpose | To guide the development of module-level programs |
| Entry Criteria | * Problem description [[1]](#footnote-1) * Project Plan Summary form [[2]](#footnote-2) * Time and Defect Recording logs 2 * Defect Type and Size Counting Standard 1 |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Planning  (see detailed script) | * Produce or obtain a requirements statement. * Estimate the size of the program. * Estimate the required development time. * Enter the estimated size and time in the Project Plan Summary form. * Complete the Time Recording log. |
| 2 | Development  (see detailed script) | * Design the program. * Implement the design. * Test the program, and fix and log all defects found. * Complete the Time Recording log. |
| 3 | Postmortem  (see detailed script) | * Complete the Project Plan Summary form with actual time, defect, and size data. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program * Completed Project Plan Summary form with estimated and actual data * Completed Time and Defect Recording logs |

## SDP1 Planning Script

|  |  |
| --- | --- |
| Purpose | To guide the SDP1 planning process |
| Entry Criteria | * Problem description * Project Plan Summary form * Time Recording log |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Program  Requirements | * Produce or obtain a requirements statement for the program. * Ensure that the requirements statement is clear and unambiguous. * Resolve any questions. |
| 2 | Size  Estimate | * Make your best estimate of the size of this program (excluding test code), in source lines of code (LOC), according to the Size Counting standard. * Enter the estimated size in the Project Plan Summary form. |
| 3 | Time  Estimate | * Make your best estimate of the time required to develop this program. * Enter the estimated time in the Project Plan Summary form. |

|  |  |
| --- | --- |
| Exit Criteria | * Documented requirements statement * Completed Project Plan Summary form with estimated program size and development time * Completed Time Recording log |

## SDP1 Development Script

|  |  |
| --- | --- |
| Purpose | To guide the development of small programs |
| Entry Criteria | * Requirements statement * Project Plan Summary form with estimated size and development time * Time and Defect Recording logs * Defect Type standard |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Design | * Review the requirements and produce a design to meet them.   + input/output formats   + the program structure (classes, responsibilities, and interactions)   + any special algorithms and data structures needed * Record in the Defect Recording log any requirements defects found. * Record time in the Time Recording log. |
| 2 | Code | * Implement the design. * Record in the Defect Recording log any requirements or design defects found. * Record time in the Time Recording log. |
| 3 | Test | * Test until all tests run without error. * Fix all defects found. * Record defects in the Defect Recording log. * Record time in the Time Recording log **(the time spent in creating tests, executing, as well as fixing defects found in test is recorded as test time)**. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program * Completed Time and Defect Recording logs |

Notes:

* You need not to follow these steps in a single sequence. You can follow them in iterations (e.g., one iteration for each class or group of classes). The important is not to skip or mix steps.

## SDP1 Postmortem Script

|  |  |
| --- | --- |
| Purpose | To guide the SDP1 postmortem process |
| Entry Criteria | * Problem description and requirements statement * Project Plan Summary form withdevelopment timedata * Completed Time and Defect Recording logs * A tested and running program. |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Defects | * Review the Defect Recording log to verify that all of the defects found in each phase were recorded. Using your best recollection, record any omitted defects. * Check that the data on every defect in the Defect Recording log are accurate and complete.Using your best recollection, correct any missing or incorrect defect data. |
| 2 | Size | * Use your LOCCounter program to determine the total size of your program, in LOC (ignoring test code!). * Enter this data in the Project Plan Summary form. |
| 3 | Time | * Review the completed Time Recording log for errors or omissions. * Using your best recollection, correct any missing or incomplete time data. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program * Completed Project Plan Summary form * Completed Time and Defect Recording logs |

# SDP1 Process forms and standards

## Project Plan Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Student(s) |  | Class |  |
| Program |  | Language |  |
| Instructor |  | Start Date |  |
|  |  | End Date |  |

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Estimated Value** | **Actual Value** |
| Size (LOC) |  |  |
| Time (min.) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Time in Phase** | **Defects Injected** | **Defects Removed** |
| PLAN - Planning |  |  |  |
| DLD - Detailed Design |  |  |  |
| CODE - Code |  |  |  |
| UT - Unit Test |  |  |  |
| PM - Postmortem |  |  |  |
| **Total** |  |  |  |

|  |  |  |
| --- | --- | --- |
| Instructions | |  |
| Purpose | To hold a summary of planned and actual data for a development project. | |
| Header | * Enter your name(s) and class, start date and end date. * Enter the program number and name. * Enter the instructor’s name and the programming language you are using. | |
| Estimated Size | * Enter the estimated size of your program, in source lines of code, **not considering blank lines, comment lines and test code** (see Size Counting Standard). | |
| Estimated Time | * Enter the estimated total development time, in minutes. | |
| Actual Size | * Enter the actual program size, **not considering blank lines, comment lines and test code, using your own LOCCounter program.** | |
| Actual Time | * Actual development time; calculated automatically based on the Time Recording Log. | |
| Time in Phase | * Shows the actual time by phase and the total time, in minutes. * It is calculated automatically based on the Time Recording Log. | |
| Defects Injected | * Shows the number of (known) defects injected by phase and the total defects injected. * It is calculated automatically based on the Defect Recording Log. | |
| Defects Removed | * Shows the number of defects removed by phase and the total defects removed. * It is calculated automatically based on the Defect Recording Log. | |

## Time Recording Log

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Start Date and Time** | **Interrup.** | **Stop Date and Time** | **Delta** | **Comments** |
|  |  |  |  |  |  |

Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to record the time you spend on each project activity. * You may have multiple entries per process phase. |
| General | * Record all of the time you spend on the project. * Record the time in minutes. * Be as accurate as possible. * If you forget to record the starting, stopping, or interruption time for an activity, promptly enter your best estimate. |
| Phase | Enter the name of the phase for the activity you are working on, e.g. PLAN.  Tool hint: select from the dropdown list. |
| Start Date and Time | Enter the date and time when you start working on a process activity.  Tool hint: double click to get the current date and time. |
| Interruption Time | * Record any interruption time that was not spent on the process activity. * If you have several interruptions, enter their total time. * You may enter the reason for the interrupt in comments. |
| Stop Date and Time | Enter the date and time when you stop working on that process activity.  Tool hint: double click to get the current date and time. |
| Delta Time | Clock time actually spent working on the process activity, less the interruption time.  Tool hint: this is calculated automatically. |
| Comments | Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity. |

## Defect Recording Log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number** | **Date** | **Type** | **Phase Injected** | **Phase Removed** | **Fix Time** | **Description** |
|  |  |  |  |  |  |  |

Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to hold data on the defects that you find and correct. |
| General | * A defect is something that has to be fixed in the code or other work product, to ensure proper operation, reuse or maintenance. * Record each defect separately and completely. |
| Number | * Enter the defect number. * For each program, use a sequential number starting with 1. |
| Date | * Enter the date on which you found the defect. |
| Type | * Enter the defect type (see Defect Type Standard). * Use your best judgment and be consistent in selecting which type applies. |
| Phase Injected | * Enter the phase when this defect was injected (e.g., CODE). * Use your best judgment. |
| Phase Removed | * Enter the phase during which you fixed the defect (e.g., UT). * This is usually the phase where the defect is found. |
| Fix Time | * Enter the time that you took to locate and fix the defect. * This time can be determined by stopwatch or by judgment. * This information is useful to determine which defects cost you most. * For simple defects that are fixed quickly, enter 1 minute. |
| Description | * Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it. |

## Defect Type Standard

|  |  |  |
| --- | --- | --- |
| **Type Number** | **Type Name** | **Description** |
| 10 | Assignment | Values assigned incorrectly or not assigned at all; incorrect type declaration. |
| 20 | Check | Incorrect data validation or incorrect conditional expression. |
| 30 | Algorithm | Efficiency or correctness problem that can be fixed without the need to change the overall program structure. |
| 40 | Interface | Incorrect interaction among classes/methods/objects/etc. (e.g., wrong method called, parameters passed by incorrect order or incorrect type). |
| 50 | Function | Significant software capability that is missing or incorrectly built, involving a considerable amount of code. |
| 60 | Timing | Incorrect temporal coordination in using shared resources. |
| 70 | Documentation | Erroneous or missing comments affecting program reuse or maintenance. |
| 80 | Environment | E.g., problems in configuration information. |

Note: For more information, see the defect types in <http://www.research.ibm.com/softeng/ODC/DETODC.HTM>.

## Size Counting Standard

* The size counting tool counts physical source lines of code, ignoring blank lines and comments.
* The ensure consistency and comparability, you should write each instruction in a separate line. This way, the physical count will equal the logical count.

# General instructions

## Assignment package

The assignment package (ESOF-Prog1.zip), downloadable from the course Web page by enrolled students, contains the following files:

* ESOF-Prog1.pdf - this document, with the assignment instructions;
* ESOF-Prog1.xls - Excel workbook for recording data about your work;

## Work alone or in pairs

You can work alone or in pairs. In case you work as a pair, the pair must be the same for the 4 assignments, and you must do the complete work together, on the same tasks (you cannot split tasks); you should also record the time as of a single person.

## Follow the process

For each process script,

* ensure that you have all of the required inputs, according to the entry conditions;
* start a new Time Recording Log entry, with the start date and time;
* execute the process steps;
* check that the exit criteria are met;
* record the stop date and time in the Time Recording Log.

In the case of the Development script, create separate time records for the different phases (Design, Code, Test).

## Review you assignment

Use the attached grading checklist to check your assignment. Ensure that your assignment is correct before you submit it. Your process data must be accurate, precise and self-consistent.

You are evaluated based on the accuracy of your data, and not on the number of defects you injected or the time you spent. This is not a programming course!

By completing this assignment you gain **1 value out of 20**.

## Submit you assignment

When you’ve completed your review, submit your assignment in a zip file (through the web page, or through an email to the instructor with “ESOF-Prog1” in the subject).

• workbook with your process data

• source program listing

• test results (e.g., print screen of test execution results)

You should complete and submit this assignment before the next class.

## Suggestions

Keep your programs simple. You will learn as much from developing small programs as from large ones. If you are not sure about something, ask your instructor for clarification.

# Grading checklist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Legend |  |  |  |  |
| √ - O.K. |  |  |  |  |
| X – Not Ok |  |  |  |  |
|  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | **Assignment Package** | **Comments** |
|  | Excel workbook (with project summary, time log and defect log) |  |
|  | Source program listing |  |
|  | Test results (e.g., print screen of test execution results) |  |

|  |  |  |
| --- | --- | --- |
|  | **Program and Test Results** | **Comments** |
|  | The program appears to be workable. |  |
|  | All required tests have been run. |  |
|  | The actual output is correct for each test. |  |

|  |  |  |
| --- | --- | --- |
|  | **Time Log** | **Comments** |
|  | Time data are entered for all process phases. |  |
|  | Process phases are sequenced appropriately. |  |
|  | Time data are entered against the appropriate process phase. |  |
|  | Time data are complete and reasonable. |  |
|  | Times were recorded as the work was done. |  |

|  |  |  |
| --- | --- | --- |
|  | **Defect Log** | **Comments** |
|  | Every defect has all the required data. |  |
|  | Defects were injected before removed. |  |
|  | Defects are adequately described. |  |
|  | Defect types are consistent with description. |  |
|  | Defect types are assigned consistently. |  |

|  |  |  |
| --- | --- | --- |
|  | **Planning Summary** | **Comments** |
|  | Estimate development time has been entered. |  |
|  | Estimated program size has been entered. |  |
|  | Actual program size has been entered correctly. |  |

|  |  |  |
| --- | --- | --- |
|  | **Consistency Checks** | **Comments** |
|  | Defects removed are consistent with time in phase and prog. size. |  |
|  | Total test defect fix times are less than test time. |  |
|  | Defect dates and phases are consistent with the time log. |  |

|  |  |  |
| --- | --- | --- |
|  | **General** | **Comments** |
|  | Followed the defined process. |  |
|  | Complete, consistent and accurate process data collected. |  |
|  | The student did his or her own work. |  |

1. In this document. [↑](#footnote-ref-1)
2. In the accompanying Excel workbook. [↑](#footnote-ref-2)