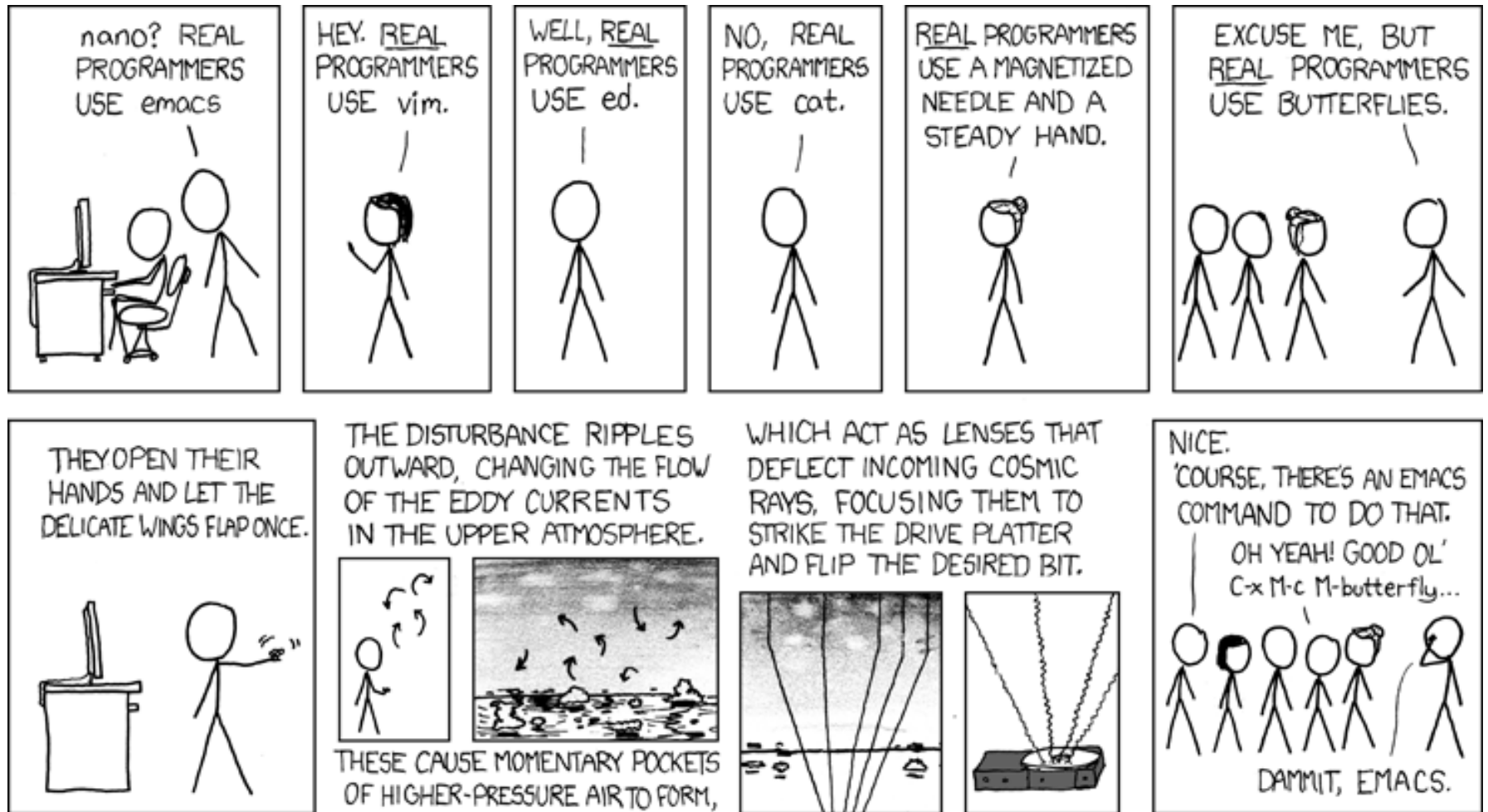


“A Magnetized Needle and a Steady Hand”

Alternatives in the modern world of Integrated
Development Environments

Jennifer Wood
CSCI 5828
Spring 2012

Real Programmers



For the rest of us

- Modern Integrated Development Environments (IDE)
 - A one-stop shop with multiple features that can be easily accessed by the developer (without switching modes or activating other utilities) to ease the task of creating software
 - A multitude of IDEs exist for each programming language (Java, C++, Python, etc.) and each platform (desktops, cell phones, web-based, etc.)
 - Some IDEs can handle multiple programming languages, but most are based in just one
 - There are many good free IDEs out there, but you can also pay for functionality from \$ to \$\$\$\$
 - IDEs are like opinions, everyone has one and everyone thinks everyone else's stinks

Why are IDEs a good thing?

- They attack many of the sources of accidental difficulties in software development by having:
 - Real-time protection from fault generating typos and bad syntax
 - High levels of abstraction to keep developers from being forced to redevelop basic (and not so basic) classes and structures for every project
 - IDE increases the power of many development tools by merging them into one that provides “integrated libraries, unified file formats, and pipes and filters. As a result, conceptual structures that in principle could always call, feed, and use one another can indeed easily do so in practice.” (Brooks, 1987).
 - A core focus of IDE developers is continuous improvement in transparency to minimize searching for functions or switching modes by the user

Why are Modern IDEs a Good Thing?

- The original IDEs were command driven and provided minimal assistance in the development of source code
 - Huge potential for making errors when coding
 - Resulted in large amounts of time spent debugging code
 - Often the debugging tools were located in a separate tool that had to be booted up by the user and run with the problem piece of code
 - Little support with the language itself to help avoid typos or bad syntax
 - You could argue that at this point accidental difficulties in software development were still running rampant

Common IDE Features

- These basics are generally included:
 - Text editor – allows writing and editing of source code
 - Build automation – generates the Makefile (or equivalent) and passes it to the compiler
 - Debugger – allows the developer to follow the execution of source code step by step to confirm correct execution or to locate faults
- These additional features are common:
 - Compiler/Interpreter
 - Error checking as you write the source code
 - Code autocompletion as you type
 - Highlighting of code syntax (including bracket matching))
 - Ability to browse and/or visualize class structures and hierarchies
 - Templates for quick creation of classes

Nice-to-Have Features

- Profiler – monitors execution of software to aid in discovery of memory leaks and performance improvement
 - Task profiling
 - Heap monitoring – look for unnecessary references and sources of memory leaks
 - Thread monitoring – check status of multiple threads
 - Identify CPU bottlenecks – methods that are monopolizing large amounts of processing time
- Database Integration – supports database development and usage
 - Often multiple database frameworks are supported (SQL, Oracle DB, Apache Derby, etc.)
- Storage Management – handles the saving and loading of multiple types of data
- Version control – keep track of what is being edited and which version is the most up to date
- UML Support – design your system with UML diagrams and use these blueprints within your IDE to build the supporting code

Nice-to-Have Features (2)

- GUI Design Tool – drag and drop design of GUI with the IDE autocoding much of the GUI framework in the source code with placeholders for user supplied methods
- Refactoring Support – assists in safe deletion of classes, encapsulation of fields (adding get/set methods), breaking a class apart into a new abstract class and subclass, renaming classes, etc.
- Testing Support
 - Code coverage
 - Unit Testing
 - Test case management
- Issue Tracking – support for maintaining a database of open issues that can be shared with the development team
- Lifecycle Integration – support for Agile tools and planning

IDEs for tomorrow, today

- Support for programs cobbled together with multiple programming languages
 - Allows developers to leverage the strengths of multiple programming languages when solving and modeling complex problems
- IDEs are available that can be run in a browser with your source code stored in the cloud
 - Supports flexible development environments where teams can share and access resources from multiple computers and locations
- IDEs are evolving to support simultaneous development for multiple frameworks and/or devices
 - Example: Your company wants you to build a new game such that can be released for desktops, tablet computers, and cell phones

Some drawbacks with modern IDEs

- Modern IDEs are complex!
 - To fully utilize the features of a big IDE takes an investment of time and training
 - There can be a lot of clutter as all the features clamor for space in the IDE workspace
 - They take up a lot more disk space and use a lot more memory than just running Vim, GCC and GDB
 - New revisions roll out regularly and features you liked can be moved or may no longer be supported
 - Example: NetBeans 7.0 has stopped supporting Ruby and Ruby on Rails (but you can get a third party plug-in)

Finding the Right IDE for Your Project (and You)

- Cost
 - There are many good free IDEs out there (Apple Xcode, Eclipse, NetBeans, etc.)
 - There are also many good IDEs that require a paid license (Microsoft Visual Studio, IBM Rational Rhapsody, etc.)
- Functionality
 - Availability of plug-ins or modules to expand capabilities
 - Multi-language support (many IDEs support a single programming language)
 - Is the platform or device you are designing for supported by the IDE?
 - Are there development team/product lifecycle supporting features?
 - Wikipedia has a nice set of basic comparison tables for many types of IDEs:
http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments
- Comfort
 - Go with what works for you
 - Get input from the people you work with and/or people in the larger community that you respect
 - Good resources for IDE selection can be places like slashdot.org, stackoverflow.com, etc.
- Reality
 - What will your company's IT department actually let you install on your machine?

Looking at NetBeans:

An Example of a Full-Featured IDE

- Supports Windows, Mac, Linux and Solaris
- Design and build web, desktop and mobile applications in Java
- Multilanguage support for Java, PHP, JavaScript, Ajax, Groovy and Grails, and C/C++
 - Ruby/Ruby on Rails no longer directly supported, but a plug-in is available at <http://plugins.netbeans.org/plugin/38549/ruby-and-rails>
- Includes some nice-to-have features:
 - Drag-and-drop GUI Design Tool (now with Visual Debugging!)
 - Supports Java Enterprise and web application development for multiple frameworks
 - Database Explorer to connect to, search, create or delete any relational database for which there is a JDBC driver
 - Version Control with Concurrent Versions System (CVS) including version integration
 - Agile friendly tools like Team Project hosting, issue tracking, and continuous build support

NetBeans – A Brief Tour: The Workspace

ConcurrentTwit - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

96.9/124.3MB

Projects: CoinTossing, ConcurrentTwit (Source Packages, `ConcurrentTwit.java`, `CreateTweetFile.java`, `ProcessTweeters.java`, `TweetCompiler.java`, `TwitGet.java`), Libraries (ConcurrentTwitter, ContactEditor, DataSourcePlotter, Exam2, Factorial, GuessingGame, JavaApplication15, TurtleGraphics)

Services

Files

ConcurrentTwit.java ProcessTweeters.java TweetCompiler.java TwitGet.java CreateTweetFile.java

```
14  */
15  public class TweetCompiler implements Runnable
16  {
17      public final String twitID;
18
19      public TweetCompiler(final String userID)
20      {
21
22
23
24
25
26
27
28
29      private ArrayList<String> allTweets = new ArrayList<String>();
30
31      for(int i =1; i<=16; i++)
32      {
33          allTweets.addAll(TwitGet.getTweets(twitID, i));
34
35      }
36
37      System.out.println("    Compiled page "+i+ " for "+twitID+". Array is now "+allTweets.size());
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Members View

- TweetCompiler :: Runnable
 - twitID : String
 - TweetCompiler(String userID)
 - run()
 - clone() : Object
 - equals(Object o) : boolean
 - finalize()
 - getClass() : Class<?>
 - hashCode() : int
 - notify()
 - notifyAll()
 - toString() : String
 - wait(long l)
 - wait(long l, int i)
 - wait()

Tasks

```
compile:
Created dir: C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\Conc
Copying 1 file to C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03
Nothing to copy.
Building jar: C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\Conc
To run this application from the command line without Ant, try:
java -jar "C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\Conc
jar:
BUILD SUCCESSFUL (total time: 1 second)
```

Here's a screen shot of my NetBeans workspace. See how much functionality is packed into a single view?

Here we can navigate the source directories and libraries of multiple projects.

Here's the built-in text editor we talked about. Note the highlighting of code syntax.

In this frame we can track the members of the class we're currently working in, including those that were inherited by the current class.

Down here we can keep track of the output of our compiler and the execution of our program. This is also where the debugger console and watch window are located (more on these later).

NetBeans – A Brief Tour: The Workspace (2)

The screenshot shows the NetBeans IDE workspace with several callouts pointing to specific buttons in the toolbar:

- Click here to send the project to javac, the Java compiler**: Points to the 'Compile' button (a hammer icon).
- Click here to attach and run the debugger**: Points to the 'Attach Debugger' button (a bug icon).
- Click here to execute the compiled code**: Points to the 'Run' button (a green play button).
- Click here to attach and run the profiler**: Points to the 'Attach Profiler' button (a clock icon).

Additional callouts provide context:

- See the buttons for commonly executed tasks? There was a time when all this functionality would have required the user to open multiple applications on their desktop.**: Points to the entire toolbar area.

The IDE interface includes a Project Explorer on the left showing the project structure, a central code editor with Java code for `TweetCompiler`, a Members View at the bottom left, and an Output window at the bottom right showing the compilation process.

```
public class TweetCompiler implements Runnable
{
    public final String twitID;

    public TweetCompiler(final String userID)
    {
        twitID = userID;
    }

    public
    {
        ArrayList<String> allTweets = new ArrayList<String>();
        // ...
    }
}
```

```
Output - ConcurrentTwit (clean.jar)
Tasks
compile:
Created dir: C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\ConcurrentTwit\dist
Copying 1 file to C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\ConcurrentTwit\build
Nothing to copy.
Building jar: C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\ConcurrentTwit\dist\ConcurrentTwit.jar
To run this application from the command line without Ant, try:
java -jar "C:\Users\spiggott\Desktop\Jen Stuff\CSCI 5828\HW03\ConcurrentTwit\dist\ConcurrentTwit.jar"
jar:
BUILD SUCCESSFUL (total time: 1 second)
```

NetBeans – A Brief Tour: Refactoring

The screenshot shows the NetBeans IDE interface with the Refactor menu open. The menu items are: Rename... (Ctrl+R), Move... (Ctrl+M), Copy..., Safely Delete... (Alt+Delete), Change Method Parameters..., Pull Up..., Push Down..., Extract Interface..., Extract Superclass..., Use Supertype Where Possible..., Move Inner to Outer Level..., Convert Anonymous to Member..., Introduce Variable... (Alt+Shift+V), Introduce Constant... (Alt+Shift+C), Introduce Field... (Alt+Shift+E), Introduce Method... (Alt+Shift+M), Encapsulate Fields..., Undo, and Redo. Blue arrows point from callout boxes to these menu items.

You can rename your class, or move your class to a different package or into another class, or copy your class – all source code will be updated to reference the new implementation.

Safely delete checks all your source code and removes the piece of code provided no other code references it.

You can change the access modifier of a method or add parameters.

Select methods from your class and create an interface OR select methods and fields to make a new superclass.

Choose a field, hit encapsulate, and have the IDE generate a template set and get method for that field.

There are lots of helpful tools for refactoring to make it as fast and easy (and error free!) as possible.

NetBeans' wiki provided these links to some great resources for understanding and implementing refactoring of your source code:

- <http://refactoring.com>
- <http://refactoring.info>
- <http://www.sourcemaking.com>

NetBeans – A Brief Tour: Debugging

The screenshot displays the NetBeans IDE interface. On the left, the Project and Navigator views show a project named 'ConcurrentTwit' with source files like 'ConcurrentTwit.java', 'CreateTweetFile.java', 'ProcessTweeters.java', 'TweetCompiler.java', and 'TwitGet.java'. The main editor window shows the 'run()' method of 'ConcurrentTwit.java'. A red line is drawn across line 42, which contains the statement `System.out.println("File complete for "+twitID);`. A blue arrow points to this line with the label 'Line Breakpoint'. The bottom of the IDE shows the 'VM Telemetry Overview' panel with the text '<no tasks>'. A yellow callout box in the top right corner explains the debugger's integration.

NetBeans' Debugger is another important integrated tool that you can set-up and execute from the main workspace.

Set a breakpoint (a place in the source code you want the debugger to stop at during execution) by clicking one the line number containing the statement you want to stop at.

NetBeans – A Brief Tour: Debugging

(2)

Keep an eye on the value of a variable during execution by adding a watch. An easy way to do this is to highlight the variable or expression you want to watch, right click on the highlighted text, and select New Watch from the pop-up menu.

You can track your Watches in the Watches window by selecting Windows>Debugging>Watches. The Watches window will be available in the bottom right frame of the workspace.

```
27 ArrayList<String> allTweets = new ArrayList<String>();
29
30 for(int i = 1; i <= 16; i++)
31 {
32     // ...
33     // ...
34     // ...
35     // ...
36 }
37
38 Create output file
39 output
40 output.addAddresses(allTweets);
41 output.closeFile();
42
43 System.out.println("File complete for "+tweetID);
44
45
46 }
47
48 }
49
```

New Watch

Watch Expression:

i

Name	Type	Value
i		
<Enter new watch>		

NetBeans – A Brief Tour: Debugging

(3)

When you have your breakpoints and watches set and you are ready to run your program, right-click on the name of your project and select Debug from the menu. Let the debugging commence...

Let's run our concurrent Tweet collecting program through the debugger so we can see some handy tools for multithreading...

```
21 }
22
23
24 public void run()
25 {
26
27     ArrayList<String> allTweets = new ArrayList<String>();
28
29     for(int i =1; i<=16; i++)
30     {
31         allTweets.addAll(TwitGet.getTweets(tweetID));
32     }
33
34
35
36
37
38
39
40
41
42
43
44     System.out.println("File complete for "+tweetID);
45
46 }
47
48 }
49
```

Watches

Name
<Enter new watch>

Output

```
File complete for 1
File complete for 2
File complete for 3
File complete for 4
File complete for 5
File complete for 6
File complete for 7
File complete for 8
File complete for 9
File complete for 10
File complete for 11
File complete for 12
File complete for 13
File complete for 14
File complete for 15
File complete for 16
```

NetBeans – A Brief Tour: Debugging

(1)

ConcurrentTwit - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

66.0/107.9MB

Projects Files Services Profiler

Keep-Alive-Timer running
'pool-1-thread-1' running
'pool-1-thread-2' running
'pool-1-thread-3' running
'pool-1-thread-4' running
'pool-1-thread-5' running
'pool-1-thread-6' running
'pool-1-thread-7' running
'pool-1-thread-8' running
'pool-1-thread-9' running

Look! We can keep track of all the threads our concurrent program has created while we debug!

You can even pause a thread in its execution – just hit the pause button.

```
17 public final String twitID;  
18  
19 public TweetCompiler(final String userID)  
20 {  
21     twitID = userID;  
22 }  
23  
24 public void  
25 {  
26  
27  
28     ArrayList<String> allTweets = new ArrayList<String>();  
29  
30     for(int  
31     {  
32         all  
33  
34         System.out.println("Compiled page " + i + " for " + twitID + " ready to now " + allTweets.size());  
35  
36     }  
37  
38     CreateTweetFile output = new CreateTweetFile();  
39     output.openFile(twitID);  
40     output.addTweets(allTweets);  
41     output.closeFile();  
42  
43     System.out.println("File complete for " + twitID);  
44  
45  
46 }  
47  
48 }  
49
```

ConcurrentTwit - Navigator

<No View Available>

Variables Breakpoints VM Telemetry Overview Output Tasks

Name	Type	Value
<Enter new watch>		

No variables to display because there is no current thread.

ConcurrentTwit (debug) 2 | 43 | 1 INS

NetBeans – A Brief Tour: Debugging

(5)

ConcurrentTwit - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

132.3/177.5MB

Projects: Files: Services: Profiler

Keep-Alive-SocketCleaner running

'pool-1-thread-1' at line breakpoint TweetCompiler.;

TweetCompiler.run:43

Hidden Source Calls

'pool-1-thread-2' at line breakpoint TweetCompiler.;

TweetCompiler.run:43

Hidden Source Calls

'pool-1-thread-3' at line breakpoint TweetCompiler.;

TweetCompiler.run:43

Hidden Source Calls

'pool-1-thread-4' running

'pool-1-thread-5' running

'pool-1-thread-6' running

'pool-1-thread-7' suspended at 'SocketInputStream'

New Breakpoint Hits (3)

Navigator

Members View

TweetCompiler :: Runnable

twitID : String

TweetCompiler(String userID)

run() Object

clone() Object

equals(Object obj) boolean

finalize() void

getClass() Class

hashCode() int

notify() void

notifyAll() void

toString() String

wait(long l) void

wait(long l, int i) void

wait() void

```
17 public final String twitID;
18
19 public TweetCompiler(final String userID)
20 {
21     twitID = userID;
22 }
23
24 public void
25 {
26
27
28     ArrayList
29
30     for(int i =1; i<=16; i++)
31     {
32         allTweets.addAll(TwitGet.getTweets(twitID, i));
33
34         System.out.println("    Compiled page "+i+ " for "+twitID+". Array is now "+allTweets.size());
35
36     }
37
38     Cre
39     outp
40     output.addAll(allTweets);
41     output.closeFile();
42
43     System.out.println("File complete for "+twitID);
44
```

Several of our threads have hit the breakpoint we set. The threads are paused and waiting for us to step through the breakpoint for that thread.

NetBeans is keeping a tally of how many breakpoints have been reached and are awaiting attention.

Thread 7 is still paused from earlier. It won't reach the breakpoint until after we start it running again. We'll set it to run again by pushing the play button.

Variables Breakpoints VM Telemetry Overview Output Tasks

Name	Type	Value
<Enter new watch>		
this	TweetCompiler	#780

ConcurrentTwit (debug) 2 | 43 | 1 INS

NetBeans – A Brief Tour: Debugging

(6)

The screenshot shows the NetBeans IDE interface with the following components and callouts:

- Toolbar:** Contains buttons for Stop (red square), Pause (two vertical bars), Play (green triangle), Step Over (orange arrow), Step Over Expression (orange arrow with 'E'), Step Into (orange arrow with 'I'), Step Out (orange arrow with 'O'), and Run to Cursor (orange arrow with 'C').
- Code Editor:** Displays the source code for `TweetCompiler.java`. A breakpoint is set at line 43. The code includes:

```
final String tweetID;  
TweetCompiler(final String userID)  
{  
    tweetID = userID;  
    ...  
    ArrayList<String> allTweets = new ArrayList<String>();  
    ...  
    TweetFile();  
}
```
- Members View:** Shows the class structure for `TweetCompiler` with methods like `run()`, `done()`, `equals()`, `finalize()`, `getClass()`, `hashCode()`, `notify()`, `notifyAll()`, `toString()`, `wait(long l)`, `wait(long l, int i)`, and `wait()`.
- Output Window:** Shows the following log messages:

```
LineBreakpoint TweetCompiler.java : 43 successfully submitted.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-8.  
Thread pool-1-thread-8 stopped at TweetCompiler.java:43.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-2.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-3.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-9.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-1.  
User program running  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-5.  
Thread pool-1-thread-5 stopped at TweetCompiler.java:43.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-6.  
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-4.
```

Callout 1: Stop, pause, or continue (play) the debugger

Callout 2: "Step over" – evaluate the current line and proceed to the next line in the file

Callout 3: "Step over expression" – evaluate the current expression and proceed to the next expression in the line

Callout 4: "Step into" – evaluate the current line AND if there is a method call in that line the debugger will take you through the lines of that method's code

Callout 5: "Run to cursor" – evaluates the code until it gets to the line your cursor is on and stops again

Callout 6: When you've reached a breakpoint, NetBeans stops the execution of your code and you can use these handy buttons to choose how to proceed.

Callout 7: "Step out" – evaluates the rest of the current method and takes you back out to the next line in the calling code

NetBeans – A Brief Tour: Debugging

(7)

The screenshot displays the NetBeans IDE interface. The main editor window shows the source code of `TweetCompiler.java` with a breakpoint set at line 43. The code includes a `run()` method that creates an `ArrayList` and iterates over it, adding tweets. The `Debugger Console` at the bottom shows the output of the program, including the message `LineBreakpoint TweetCompiler.java : 43 successfully submitted.` and several `Breakpoint hit` messages for different threads. A yellow callout box with a blue arrow pointing to the `Debugger Console` tab contains the text: "You can watch the output from the debugger down in the Debugger Console."

```
public final String twitID;

public TweetCompiler(final String userID)
{
    twitID = userID;
}

public void run()
{
    ArrayList<String> allTweets = new ArrayList<String>();

    for(int i =1; i<=16; i++)
    {
        allTweets.addAll(TwitGet.getTweets(twitID, i));

        System.out.println("    Compiled page "+i+" for "+twitID+". Array is now "+allTweets.size());
    }
}

CreateTweetFile
output.openFile(
output.addTweets
output.closeFile
```

Debugger Console

```
User program running
LineBreakpoint TweetCompiler.java : 43 successfully submitted.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-8.
Thread pool-1-thread-8 stopped at TweetCompiler.java:43.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-2.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-3.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-9.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-1.
User program running
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-5.
Thread pool-1-thread-5 stopped at TweetCompiler.java:43.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-6.
Breakpoint hit at line 43 in class TweetCompiler by thread pool-1-thread-4.
```

NetBeans – A Brief Tour: Debugging

(10)

ConcurrentTwit - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

142.3/177.5MB

Projects | Files | Services | Profiler

Start Page | ConcurrentTwit.java | ProcessTweeters.java | TweetCompiler.java | TwitGet.java | CreateTweetFile.java

Keep-Alive-SocketCleaner' running

- 'pool-1-thread-4' at line breakpoint TweetCompiler.java:43
 - TweetCompiler.run:43
 - Hidden Source Calls
- 'pool-1-thread-5' at line breakpoint TweetCompiler.java:43
 - TweetCompiler.run:43**
 - Hidden Source Calls
- 'pool-1-thread-6' at line breakpoint TweetCompiler.java:43
 - TweetCompiler.run:43
 - Hidden Source Calls
- 'pool-1-thread-7' suspended at 'SocketInputStream.soc

New Breakpoint Hits (2)

Navigator

Members View

TweetCompiler :: Runnable

- twitID : String
- TweetCompiler(String userID)
- run()
- clone() : Object
- equals(Object o) : boolean
- finalize()
- getClass() : Class<?>
- hashCode() : int
- notify()
- notifyAll()
- toString() : String
- wait(long l)
- wait(long l, int i)
- wait()

```
17 public final String twitID;
18
19 public TweetCompiler(final String userID)
20 {
21     twitID = userID;
22 }
23
24 public void run()
25 {
26
27
28     ArrayList<String> allTweets = new ArrayList<String>();
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

Variables

ConcurrentT...

Could not ...

Compile ...

Retrieved p...

Compiled page 13 for planetmoney. Array is now 33100

Retrieved page 14 for planetmoney

Compiled page 14 for planetmoney. Array is now 36148

Retrieved page 15 for planetmoney

Compiled page 15 for planetmoney. Array is now 39173

File complete for birbigis

Retrieved page 16 for pl

Compiled page 16 for

File complete for totm

File complete for azizan

See how the number of threads has dropped off as tweet collection tasks have completed? These four threads from the executor service pool that are left are all stuck at the breakpoint waiting.

Once they are released, the program will run through the rest of its execution. (In case you're wondering, the other five threads in the pool were already released from our breakpoint.)

Next, let's take a look at the Profiler tool using our Concurrent Twitter program...

ConcurrentTwit (debug) | running... | 2 | 43 | 1 | INS

NetBeans – A Brief Tour: Profiler

Click here to run the profiler

The Profiler Setup will open.

If you want to collect more information on what the individual threads are doing check "Enable Thread Monitoring" in the Monitor tab.

```
14  */
15  public class TweetCompiler implements Runnable
16  {
17      public final String tweetID;
18
19      public TweetCompiler(final String userID)
20  {
```

is now "+allTweets.size());

Advanced settings

Run Cancel Help

28\HW03\ConcurrentTwit\dist\ConcurrentTwit.jar
try:
HW03\ConcurrentTwit\dist\ConcurrentTwit.jar"

NetBeans – A Brief Tour: Profiler (2)

In this view we're looking at processing time per method per thread. You can see that most of the time (99.7%) of each of the executor service created threads is spent in the getTweets method where all the I/O with Twitter occurs. This is what we expected when we built the program.

Thread	Method	Time (ms)	Percentage	Count
pool-1-thread-1	TweetCompiler.run ()	45978	100%	1
	TwitGet.getTweets (String, int)	45978	100%	1
	CreateTweetFile.addTweets (java.util.ArrayList)	45862	99.7%	16
	Self time	101	0.2%	1
	CreateTweetFile.openFile (String)	11.9	0%	1
	CreateTweetFile.closeFile ()	1.70	0%	1
pool-1-thread-8	TweetCompiler.run ()	43417	100%	1
	TwitGet.getTweets (String, int)	43417	100%	1
	CreateTweetFile.addTweets (java.util.ArrayList)	43236	99.6%	16
	Self time	173	0.4%	1
	CreateTweetFile.openFile (String)	4.61	0%	1
	CreateTweetFile.closeFile ()	1.75	0%	1
pool-1-thread-3	TweetCompiler.run ()	42959	100%	1
	TwitGet.getTweets (String, int)	42959	100%	1
	CreateTweetFile.addTweets (java.util.ArrayList)	42847	99.7%	16
	Self time	104	0.2%	1
	CreateTweetFile.openFile (String)	4.88	0%	1
	CreateTweetFile.closeFile ()	1.67	0%	1
pool-1-thread-4	TweetCompiler.run ()	40515	100%	1
	TwitGet.getTweets (String, int)	40515	100%	1
	CreateTweetFile.addTweets (java.util.ArrayList)	40236	99.7%	16
	Self time	0.236	0%	1
	CreateTweetFile.openFile (String)	0.002	0%	1
	CreateTweetFile.closeFile ()	0.002	0%	1

After the tweet collection finishes, NetBeans asks if you want to save a snapshot of the profiled execution. Click yes and then you can examine how much time each thread is spending in what methods, classes, or packages.

NetBeans – A Brief Tour: Profiler (3)

NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Files Services Prof... 126.2/202.3MB

View

VM Telemetry Threads

Basic Telemetry

Instrumented:

Filter:

Threads:

Total Memory:

Used Memory:

Time Spent in GC:

Threads

0:00 0:30 1:00 [m:s]

Attach Listener

Reference Handler

Finalizer

Signal Dispatcher

main

pool-1-thread-7

pool-1-thread-3

pool-1-thread-9

pool-1-thread-4

pool-1-thread-2

pool-1-thread-8

pool-1-thread-5

DestroyJavaVM

pool-1-thread-1

pool-1-thread-6

Keep-Alive-Timer

Keep-Alive-SocketCleaner

Keep-Alive-Timer

Running Sleeping Wait Monitor

Timeline Table Details

VM Telemetry Overview Output - ConcurrentTwit (profile) Tasks

<No View Available>

You can look at detailed information on the threads used by the application by clicking View > Threads in the Profiler window.

Here you can see the timeline of the pool of threads created by the concurrent Twitter program running.

NetBeans – A Brief Tour: Profiler (4)

NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects Files Services Prof... 146.3/202.3MB

Show: All Threads

Thread	Running	Sleeping	Wait	Monitor	Total
Attach Listener	46.910 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	46.910
Signal Dispatcher	46.910 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	46.910
pool-1-thread-9	42.510 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	42.510
DestroyJavaVM	42.510 (90.6%)	0.0 (0.0%)	4.400 (9.3%)	0.0 (0.0%)	46.910
pool-1-thread-7	41.509 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	41.509
pool-1-thread-5	39.809 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	39.809
pool-1-thread-6	38.609 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	38.609
pool-1-thread-2	38.109 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	38.109
pool-1-thread-1	38.109 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	38.109
pool-1-thread-4	37.409 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	37.409
pool-1-thread-8	37.109 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	37.109
pool-1-thread-3	32.808 (99.6%)	0.0 (0.0%)	0.100 (0.3%)	0.0 (0.0%)	32.908
Keep-Alive-SocketCleaner	5.000 (11.3%)	0.0 (0.0%)	39.070 (88.6%)	0.0 (0.0%)	44.070
Reference Handler	0.200 (0.4%)	0.0 (0.0%)	46.609 (99.3%)	0.101 (0.2%)	46.910
Finalizer	0.101 (0.2%)	0.0 (0.0%)	46.609 (99.3%)	0.200 (0.4%)	46.910
Keep-Alive-Timer	0.0 (0.0%)	2.600 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	2.600
Keep-Alive-Timer	0.0 (0.0%)	4.924 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	4.924
Keep-Alive-Timer	0.0 (0.0%)	5.007 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	5.007
Keep-Alive-Timer	0.0 (0.0%)	5.031 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	5.031
Keep-Alive-Timer	0.0 (0.0%)	5.000 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	5.000
Keep-Alive-Timer	0.0 (0.0%)	5.005 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	5.005
Keep-Alive-Timer	0.0 (0.0%)	5.001 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	5.001
Keep-Alive-Timer	0.0 (0.0%)	11.801 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	11.801
main	- (-%)	- (-%)	- (-%)	- (-%)	- (-%)

Timeline Table Details

VM Telemetry Overview Output - ConcurrentTwit (profile) Tasks

Compiled page 16 for nytimes. Array is now 36621
File complete for nytimes
Could not retrieve complete 15 for funnyordie

<No View Available>

By selecting the Table view, you can get a summary of how much time each thread spent Running, Sleeping, Waiting or Monitoring.

NetBeans – A Brief Tour: Profiler (5)

NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects Files Services Prof...
Delete
Save As...
Load...

View
VM Telemetry Threads

Basic Telemetry
Instrumented:
Filter:
Threads:
Total Memory:
Used Memory:
Time Spent in GC:

...ava ProcessTweeters.java TweetCompiler.java TwitGet.java CreateTweetFile.java Threads VM Telemetry CPU: 10:59:00 PM *
Show: All Threads

pool-1-thread-7 [java.lang.Thread] (finished) Hide

User thread

General Details

Running 0:41.509 (100.0%)
Sleeping 0:00.000 (0.0%)
Wait 0:00.000 (0.0%)
Monitor 0:00.000 (0.0%)
Total: 0:41.509

0:00 0:10 0:20 0:30 0:40 [ms]

pool-1-thread-3 [java.lang.Thread] (finished) Hide

User thread

General Details

Running 0:32.808 (99.6%)
Sleeping 0:00.000 (0.0%)
Wait 0:00.100 (0.3%)
Monitor 0:00.000 (0.0%)
Total: 0:32.908

0:00

pool-1-thread-9 [java.lar

Timeline Table Details

VM Telemetry Overview Output - ConcurrentTwit (profile) Tasks

Compiled page 16 for nytimes. Array is now 36621
File complete for nytimes
Could not retrieve complete 15 for funnyordie

<No View Available>

By selecting the Details view of the Threads tab, you can look at each thread individually and even get timestamps for the thread states.

NetBeans – A Brief Tour: Profiler (6)

By selecting the Memory (Heap) view of the VM Telemetry tab, you can look at how much memory was allocated at any time during the execution and how much was actually used.

10:58:40.617 PM, Mar 19, 2012
Heap Size 108,044,288 B
Used Heap 53,100,000 B
Max Heap Size 108,044,288 B
Max Used Heap 91,253,952 B

Memory (Heap) | Memory (GC) | Threads / Loaded Classes

VM Telemetry Overview | Output - ConcurrentTwit (profile) | Tasks

Compiled page 16 for NewYorker. Array is now 35725
File complete for NewYorker

You can look at high-level information on memory usage, garbage collection, and thread activity in the Virtual Machine Telemetry tab by the application by clicking View>VM Telemetry in the Profiler window.

NetBeans – A Brief Tour: Profiler (7)

The screenshot displays the NetBeans IDE 7.0.1 interface with the Profiler tool active. The main window shows the 'VM Telemetry Overview' section, which contains three graphs: 'Memory (Heap)', 'Surviving Generations', and 'Threads'. The 'Memory (Heap)' graph is a stacked area chart showing 'Heap Size' (red) and 'Used Heap' (purple) over time. The 'Surviving Generations' graph is a step chart showing the number of surviving generations (red) and the 'Relative Time Spent in GC' (purple). The 'Threads' graph is a line chart showing the number of 'Threads' (red) and 'Loaded Classes' (purple) over time. A tooltip is visible on the right side of the 'Threads' graph, showing the following data:

10:58:52.659 PM, Mar 19, 2012
Threads: 8
Loaded Classes: 1,172
Max Threads: 14
Max Loaded Classes: 1,178

Callouts provide additional information:

- Memory (Heap):** You can also look at the VM Telemetry in this frame. All three views available in VM Telemetry are shown side by side.
- Surviving Generations:** Garbage collection/Surviving generations: Relative time - the percentage of the total execution time that the VM spends performing garbage collection with all threads suspended. Surviving generations – the number of different ages for all objects allocated (age is the number of garbage collections survived).
- Threads:** Threads – shows the current and maximum number of threads in program.

NetBeans – A Brief Tour: Profiler (8)

There are a couple of things about this implementation of a concurrent tweet collector that we could observe from the profiler:

- Each thread spent a lot of time waiting on I/O → should rewrite our program to dispatch more threads per our concurrency textbook's recommendations
- None of our executor service threads spend time waiting on another thread
- Linear growth on memory usage, which is what we would expect as we store pages of tweets → we don't appear to have a memory leak

Summary

- Modern Integrated Development Environments provide a high level of functionality for programmers in a unified workspace
- Reduces the need for users to switch views or open other tools while building and testing software
- Use of IDEs has made it easier to prevent accidental difficulties from swamping your development project
- A multiverse of IDEs exists allowing you to find the right one for your programming language that provides the features you need
- NetBeans, primarily used for Java applications, provides much of the support and many of the features that are typical of a modern IDE
 - Learning to use these tools takes time and effort, but can really increase your productivity and the quality of your final product

Sources

- Brooks, Frederick P., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, Vol. 20, No. 4 (April 1987) pp. 10-19.
- Eclipse Resources. Eclipse.org. Retrieved March 18, 2012 from <http://www.eclipse.org/resources/index.php>
- Integrated development environment. Wikipedia. Retrieved March 16, 2012 from http://en.wikipedia.org/wiki/Integrated_development_environment
- Intersimone, D. (2010, July 2). The evolution to integrated development environments. *Computerworld.com*. Retrieved March 19, 2012 from http://blogs.computerworld.com/16451/the_evolution_of_the_integrated_development_environment_id
- NetBeans IDE 7.1 Features. *Netbeans.org*. Retrieved March 18, 2012 from <http://netbeans.org/features/index.html>
- Nourie, D. (2005, March 24). Getting Started with an Integrated Development Environment. *Sun Developer Network (SDN)*. Retrieved March 16, 2012 from <http://java.sun.com/developer/technicalArticles/tools/intro.html>
- O'Dell, J. (2010, October 6). A Beginner's Guide to Integrated Development Environments. *Mashable.com*. Retrieved March 17, 2012 from <http://mashable.com/2010/10/06/ide-guide/>
- Refactoring. *NetBeans Wiki*. Retrieved March 20, 2012 from <http://wiki.netbeans.org/Refactoring>
- Visual Studio 2010 Comparison. *Microsoft Visual Studio*. Retrieved March 18, 2012 from <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/product-comparison>
- Walker, Kevin. (2011, November 30). Netbeans Debugger Tutorial. *Ehow.com*. Retrieved March 31, 2012 from http://www.ehow.com/way_6566313_netbeans-debugger-tutorial.html