We are built to make mistakes, coded for error.

Lewis Thomas

It is one thing to show a man that he is in error, and another to put him in possession of the truth.

John Locke



Prerequisites for Eclipse

To use Eclipse you must have an installed version of the Java Runtime Environment (JRE).

The latest version is available from **java.com**.

Since Eclipse includes its own Java compiler, it is not strictly necessary to have a version of the Java Development Kit (JDK) installed on your computer.

However, I recommend installing one anyway so that you can test your code against the "real" Java compiler.

The latest version is available from: www.oracle.com/technetwork/java/

If you install the JDK, I recommend putting it in a root-level directory, and making sure there are no spaces in the pathname for the directory.



Eclipse Workbench



The initial Eclipse Workbench (my configuration):



Eclipse Java Perspective Toolbar



- - - - -





Build Project



Start Debugging + configurations

Run Project + configurations

Run Last Tool + configurations



010

Data Structures & Algorithms

Eclipse Java Perspective Toolbar





Creating a New Java Project

In the Workbench, select File/New/Java Project:

st 🔘	va - Eclipse			
File	Edit Source Refactor Navigate Search Project Run	Wind	ow Help	
	New Alt+Shift+N >	123	Java Project	🛛 🖛 🖓 Debug 👹 Java
	Open File	C.	Project	
	Close Ctrl+W	₿	Package	8
	Close All Ctrl+Shift+W	G	Class	
	Save Ctrl+S	Ø	Interface	
	Save As	G	Enum	8
14 16 16	Save All Ctrl+Shift+S	@	Annotation	1 III III III III III III III III III I
100	Revert	₽ Ŷ	Source Folder	*
		诊	Java Working Set	
	Move	C ²	Folder	B
	Rename F2	P	File	
8	Refresh F5	Ê	Untitled Text File	
	Convert Line Delimiters To	E	JUnit Test Case	
۵	Print Ctrl+P	đ	Task	
	Switch Workspace	1	Example	
	Restart	2	Other Ctrl+N	
2	Import	Ctrl+W Project Ctrl+Shift+W Class Ctrl+Shift+W Interface Ctrl+Shift+S Interface Image: Source Folder Source Folder Source Folder Java Working Set File File Image: Source Folder Source Folder Java Working Set File File Image: Source Folder Source Folder Java Working Set File Image: Source Folder Source Folder Source Folder Image: Source Folder S		
	Export			
	Properties Alt+Enter			
	1 Log.txt [D:/JavaDevelopment/DebugList]			
2 C	2 testDriver.java [DebugList/src]			
	3 DList.java [DebugList/src/debug//DS]			🦷 🖉 🖉 🕅
	4 prQuadtree.java [P4/Major_Project/DS]			



Data Structures & Algorithms

Creating a New Java Project

In the resulting dialog box:

Enter a name for the Project.

For now, just take the defaults for the remaining options.

Click **Next** and then **Finish** in the next dialog.

roject name: DList		
ocation: D:\JavaDevelopment\DList		B <u>r</u> owse
RE		
Use an execution environment JRE:	JavaSE-1.6	•
Use a project specific JRE:	јгеб	*
Use def <u>a</u> ult JRE (currently 'jre6')		Configure JREs
Project layout		
Use project folder as root for source	es and class files	
<u>Create</u> separate folders for sources	and class files	Configure default
Norking sets		
Add project to working sets		
W <u>o</u> rking sets:		* S <u>e</u> lect



Data Structures & Algorithms

Debugging 7

Adding Source for the DList Example

Download the file **DListExample.zip** from the course website Resources page, and place the contents into the **src** directory for the Eclipse project you just created:





Performing a Build

Back in Eclipse, right-click on the project icon for **DList** and select **Refresh**...

Use the **Project** menu or click on the **Build All** button (**b**) to compile the code.





Data Structures & Algorithms

Running the Program

To execute the program, click on the Run button (2).

As indicated by the source code, the test driver writes its output to a file named **Log.txt**:

Unfortunately, there appears to be an error; the value 5 should have been added to the list and appear in the final listing of the contents... it's not there.

🔐 D:\Ja	avaDe	velopm	ent\DList\Log.txt - Notepad++
<u>File</u>	dit S	Search	View Encoding Language
a P			
-	1		
E Log	.bd		
1	0:	0	
2	1:	1	display of
3	2:	2	
4	3:	3	initial list
5	4:	4	
6	5:	5	
7	6:	6	
8	7:	7	
9	8:	8	
10	9:	9	
11			
12	0:	0	
13	1:	1	display of
14	2:	2	lict offer
15	3:	3	listaller
16	4:	4	deletina 5
17	5:	6	deleting 5
18	6:	7	
19	7:	8	
20	8:	9	
21			
22	0:	0	
23	1:	1	display of
24	2:	2	list offer
25	3:	3	list after
26	4:	4	roinsorting 5
27	5:	6	ieniseiting b
28	6:	7	
29	7:	8	
30	8:	9	

Debugging 10

length:171 lines:32 Ln:1 Col:1 Sel:0



Data Structures & Algorithms

Now, we have some clues about the error:

- The list appears to be OK after the first **for** loop completes; that doesn't indicate any problems with the **add()** method called there.
- The list appears to be OK after the call to the **removeFirstOccurrenceOf()** method; that doesn't indicate any problems there.
- The list is missing an element after the call to the second **add()** method; that seems to indicate the problem lies there...

It would be useful to be able to run the program to a certain point, check the state of the list (and perhaps other variables), and then step carefully through the subsequent execution, watching just how things change.

Fortunately, Eclipse provides considerable support for doing just that.



Kinds of Breakpoints

A *breakpoint* marks a location or condition under which we want the program's execution to be suspended.

Eclipse supports setting four kinds of breakpoints:

line breakpoint	halt when execution reaches a specific statement
method breakpoint	halt when execution enters/exits a specific method
expression breakpoint	halt when a user-defined condition becomes true, or changes value
exception breakpoint	halt when a particular Java exception occurs (caught or not)



Setting a Line Breakpoint

line breakpoint

halt when execution reaches a specific statement

To set one, just double-click in the editor margin next to the selected line of code:

Java - DList/src/testDriver.java - Eclipse	ent Dec 14		
			时 教 Debug 周Java
E §	9 ×	1 import java.10.Filewriter;	
BST BST	<u> </u>	<pre>public class testDriver {</pre>	
🔁 DList		7	
🗁 src		Be public static void main(String[] args) {	8
🕀 (default package)			80
🚺 testDriver.java		filewriter log;	*
debug.Examples.DS	1	<pre>2 log = new FileWriter("Log.txt");</pre>	
<pre>3 Java - DList/src/testDriverjava - Eclipse ile Edit Source Refactor Navigate Search Project Run Window Help T - Refactor Navigate Search Project Run Window Help Package Explorer R</pre>	3		
O DList <anytype></anytype>	14	<pre>4 DList<integer> list = new DList<integer>();</integer></integer></pre>	8
G DNode	1	5	
△ front	25		
△ rear	E 1:		
△ size	19	<pre>9 list.display(log);</pre>	
● ^C DList()	20	<pre>log.write("\n");</pre>	
add(AnyType) : void	2:	1	
add(int, AnyType) : void	2.	<pre>2 list.removeFirstOccurrenceOf(5);</pre>	
o clear() : void	2.	<pre>1 log write("\n"):</pre>	
display(FileWriter) : void	2	5	
get(int) : AnyType	2	<pre>6 list.add(5, 5);</pre>	
isEmpty() : boolean	2	<pre>7 list.display(log);</pre>	
remove(int) : AnyType	2	<pre>8 log.write("\n");</pre>	
removeFirstOccurrenceOf(Any	ype)	log_close():	
set(int, AnyType) : void	3:	1 }	
A size () int		catch (TOExcention e) {	-
	•	4	•
. □* V	ritable	Smart Insert 16:43	🐻 🛛 🕫 @ 😡 📮 💽



Data Structures & Algorithms

Running to a Breakpoint

Go to the **Run** menu and select **Debug** (or use the keyboard shortcut **F11**):

Java - DList/src/testDriver.java - Eclipse						x
File Edit Source Refactor Navigate Search Project	Run	Window Help		9		
C - B @ @ @ \$\$ - O - Q - 2 #	000 1000	Run Debug	Ctrl+F11 F11	🖓 ▼ 🌾 🗢 ▼ 🔿 ▼ 📑 🅸 Debug	Ja	ava
Package Explorer 🛛 🔋 Type Hierarchy	-				- 0	
E 😫 🕫		Run History	•		^	
t BST i DList		Run As Run Configurations	•			
 default package) testDriver.java debug.Examples.DS 		Debug History Debug As Debug Configurations	<pre>> args) { txt");</pre>		**	
 DList.java DList<anytype></anytype> DNode front rear size DList() add(AnyType) : void add(int, AnyType) : void clear() : void 	 ● ●	Toggle Breakpoint Toggle Line Breakpoint Toggle Method Breakpoint Toggle Watchpoint Skip All Breakpoints Remove All Breakpoints Add Java Exception Breakpoint Add Class Load Breakpoint	Ctrl+Shift+B	<pre>DList<integer>(); ++) { :0f(5);</integer></pre>	H.	•
 display(FileWriter) : void get(int) : AnyType isEmpty() : boolean remove(int) : AnyType removeFirstOccurrenceOf(AnyType set(int, AnyType) : void cira() : int 		All References All Instances Instance Count Watch Inspect Dieplay	Ctrl+Shift+N Ctrl+Shift+I Ctrl+Shift+D		•	
□ ◆ Writal	9	Execute	Ctrl+U	6 8 0	Q 📮	8



Debug Perspective

This opens the Debug Perspective:

Debug - DList/src/testDriver.java - Eclipse	and the second value of th	No. of Concession, Name	
<u>File Edit Source Refactor Navigate Search Project Run</u>	<u>W</u> indow <u>H</u> elp		
1 - 🛛 🗟 🚔 👘 🔅 + 🔾 - 🗞 - 🤔 🧀 🔗 -	• 🕫 📝 🐳 🔳 🔳	<mark>∲</mark> • ∰ • ⇔ • → •	🗈 🕸 Debug 🐉 Java
🚺 testDriver.java 🖾 🚺 DList.java	-	🖾 🕪= Variables 🕱	
1⊕ import java.io.FileWriter;	*	Name	Value 🔺
<pre>5 6 public class testDriver {</pre>	Source	() args	String[0] (id=16)
7 8⊖ public static void main(String[] args) {		© list	DList <anytype> (id=26)</anytype>
9 10 FileWriter log; 11 try {	E		Variables
<pre>12 log = new FileWriter("Log.txt"); 13 14 DList<integer> list = new DList<integer> list = n</integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></integer></pre>	teger>();		variables
<pre> for (int i = 0; i < 10; i++) { for (int i = 0; i < 10; i++) { list.add(i); list.display(log); list.display(log)(log); list.display(log)(log)(log)(log)(log)(log)(log)(log)</pre>			
<pre>20 log.write("\n"); 21 22 list.removeFirstOccurrenceOf(5); 33 list_display(log); 4</pre>		< III	· · · ·
Debug 🕅			
 testDriver (5) [Java Application] testDriver at localhost:62844 Thread [main] (Suspended (breakpoint at line 16 in test testDriver.main(String[]) line: 16 D:\jdk1.6.0_32\jre6\bin\javaw.exe (Sep 10, 2012 5:04:27 PM) 	stDriver)) 1)	Execution sta	ick
□*	Writable Smart I	nsert 16 : 1	6 8

You may see a different window layout; feel free to close other Views, like Outline if they are visible.



Data Structures & Algorithms

Using the Variables View

At this point, the list constructor has run... let's examine the structure:

x)= Variables 🖾	🐑 📲 🗖 🖓	3
Name	Value	*
args	String[0] (id=16)	
⊳	FileWriter (id=17)	
⊿ [©] list	DList <anytype> (id=26)</anytype>	
A front	DList\$DNode (id=28)	
🖂 🔺 rear	DList\$DNode (id=30)	
▲ size	0	
		+
٠ III	•	

Objects are assigned unique IDs as they are created; these allow us to infer the physical structure...



Using the Variables View

)= Variables 🛛		E
Name	Value	-
() args	String[0] (id=16)	
⊳ [©] log	FileWriter (id=17)	
a 🔍 list	DList <anytype> (id=26)</anytype>	
🔺 🔺 front	DList\$DNode (id=28)	
🔺 elem	null	
🔈 🔺 next	DList\$DNode (id=30)	
🔺 prev	null	
A ^F this\$0	DList <anytype> (id=26)</anytype>	
🔺 🔺 rear	DList\$DNode (id=30)	
🔺 elem	null	
🔺 next	null	
p A prev	DList\$DNode (id=28)	
A ^F thisS0	DList <anytype> (id=26)</anytype>	
🔺 size	0	
		-

OK, that looks just fine... two guard nodes pointing at each other, neither holding a data value.





Data Structures & Algorithms

The Debug Toolbar

Debugging 18



- 1. Resume Continues execution until breakpoint or thread ends
- 2. Suspend Interrupts a running thread
- 3. Terminate Ends the execution of the selected thread
- 4. Disconnect Disconnect from a remote debugging session
- 5. Remove terminated launches Closes all terminated debug sessions
- 6. Step Into Steps into a method and executes its first line of code
- 7. Step Over Executes the next line of code in the current method
- 8. Step Return Continues execution until the end of the current method (until a return)
- **9. Drop to Frame** Returns to a previous stack frame
- **10. Step with Filters** Continues execution until the next line of code which is not filtered out



Step-by-step Execution

For illustration, we'll examine the insertion of the first data node, step by step: X Debug - DList/src/testDriver.java - Eclipse File Edit Source Refactor Navigate Search Project Run Window Help 📬 = 🗟 🙆 📓 🕸 = 🔕 = 🧶 = 🤔 😂 🔗 = 🗇 🍠 🐨 🗐 🕤 🗄 = 친 = 친 = 수 = 수 = 🖹 🅸 Debug 🐉 Java - 8 🚺 testDriver.java 🛛 🚺 DList.java (x)= Variables 🔀 約 📲 🖻 8 public static void main(String[] args) { 86 Name Value 9 args String[0] (id=16) FileWriter log; 10 D O log FileWriter (id=17) 11 try { log = new FileWriter("Log.txt"); DList<AnyType> (id=26) 12 Þ Iist 13 0 i 0 DList<Integer> list = new DList<Integer>(); 14 15 for (int i = 0; i < 10; i++) { 216 17 list.add(i); 18 19 list.display(log); 20 log.write("\n"); 52 21 22 list.removeFirstOccurrenceOf(5); 23 list.display(log); 24 log.write("\n"); 25 26 list.add(5, 5); 27 list display(log). 4 ~ - -30.25 参 Debug 🖾 × 11 2.9 69 testDriver (5) [Java Application] testDriver at localhost:63043 Thread [main] (Suspended) testDriver.main(String[]) line: 17 D:\jdk1.6.0_32\jre6\bin\javaw.exe (Sep 10, 2012 5:10:37 PM) 6 8 🔎 Note the appearance of the variable i and its value.



Data Structures & Algorithms

Step-by-step Execution

CS_@VT

Click the **step-into** button again; now we'll enter the call to **add()**: Debug - DList/src/debug/Examples/DS/DList.java - Eclipse File Edit Source Refactor Navigate Search Project Run Window Help 🕸 • 🔘 • 💁 • 😕 😂 🔗 • 🗣 🌶 💀 🗐 🕥 🖞 • 🏹 • 😓 • 🔶 • 🔿 • 😰 🏂 Debug 🐉 Java - 8 約 🕫 🗖 ▽ 🗖 🗖 (x)= Variables 🔀 J testDriver.java J DList.java 🖾 8 rear.prev = front: 35 Name Value 36 size = 0; DList<AnyType> (id=26) 37 DList\$DNode (id=28) ⊿ ▲ front 38 390 public void add(AnyType elem) { A elem null -40 A next DListSDNode (id=30) 41 DNode toAdd = new DNode(elem, rear.prev, rear); null A prev 42 rear.prev.next = toAdd; A thisSO DList<AnyType> (id=26) 43 rear.prev = toAdd; DList\$DNode (id=30) 44 ++size; 🖌 🛕 rear 45 null A elem 46 null A next 470 public void add(int index, AnyType elem) throws IndexOutOfBc DList\$DNode (id=28) > A prev 48 F this\$0 DList<AnyType> (id=26) 49 if (index < 0 || index > size) throw new IndexOutOfBou A size 0 50 51 DNode successor; ⊿ ⁽⁾ elem Integer (id=31) 52 if (index == size) { value 0 53 successor = rear; 51 1 N 3. O. R 3. 7 3 × . Debug 🖾 ▲ 🗾 testDriver (5) [Java Application] ▲ 🔗 testDriver at localhost:63043 Thread [main] (Suspended) 2 DList<AnyType>.add(AnyType) line: 41 testDriver.main(String[]) line: 17 D:\idk1.6.0 32\ire6\bin\iavaw.exe (Sep 10, 2012 5:10:37 PM) 6 1 8

Now, I don't really want to trace the constructor, much less the call to **new**, so this time I'll click the **step-over** button...

Step-over versus Step-into

The difference is that if you are executing a method call (or invoking new, for example) in the current statement:

step-intotakes you into the implementation of that methodstep-overcalls the method, but does not step you through its execution

Both are useful... step-into is frustrating when system code is involved.



Step-by-step Execution

8

So, we see that the needed node has been properly initialized: Debug - DList/src/debug/Examples/DS/DList.java - Eclipse File Edit Source Refactor Navigate Search Project Run Window Help 1 - 🗟 🙆 🐞 - 🔕 - 🤹 - 🖉 🗁 🦧 - 🖗 🌶 🚽 🗊 🗍 🖞 - 친 - 누 수 - 수 -😭 🏇 Debug 🐉 Java - 8 約 41 日 7 日 日 J testDriver.java 🚺 DList.java 🖾 (X)= Variables 🔀 rear.prev = front; 35 Value Name 36 size = 0; DList<AnyType> (id=26) ▲ ● this 37 } DList\$DNode (id=28) ⊿ ▲ front 38 390 public void add(AnyType elem) { A elem null 40 DListSDNode (id=30) A next 41 DNode toAdd = new DNode(elem, rear.prev, rear); null A prev 42 rear.prev.next = toAdd; Ξ b A thisSO DList<AnyType> (id=26) 43 rear.prev = toAdd; 🔺 🛕 rear DList\$DNode (id=30) 44 ++size; 45 } null 🔺 elem 46 A next null 470 public void add(int index, AnyType elem) throws IndexOutOfBc DList\$DNode (id=28) b A prev 48 DList<AnyType> (id=26) ▷ ▲ this\$0 49 if (index < 0 || index > size) throw new IndexOutOfBou 0 A size 50 51 DNode successor; ⊿ ⁽⁾ elem Integer (id=31) 52 if (index == size) { value 0 53 successor = rear; ⊿ ⁽⁾ toAdd DList\$DNode (id=36) 54 ⊿ ▲ elem Integer (id=31) 55 else { value 0 56 successor = front.next; 57 int i = 0; DList\$DNode (id=30) A next 58 while (i < index) {</pre> b A prev DList\$DNode (id=28) 59 successor = successor.next; ▷ ▲ this\$0 DList<AnyType> (id=26)



CS_@VT

60

C1

۰.

++i;

111

Data Structures & Algorithms

1

III

M IÞ 88

- N 3. 9. 8 - 5

ь.

- -

8 1

 ∇

0

69

Step-by-step Execution

Three clicks on **step-over** (or **step-into**) bring us to this point:

le <u>E</u> dit <u>S</u> ource Refac <u>tor N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp → □ □ □ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	·····································	
1	9 · 2 · · · · · · ·	
		😭 🏂 Debug 🖏 Java
🕽 testDriver.java 🚺 DList.java 🕱 🖓 🖓	□ (×)= Variables 🛛	
35 rear.prev = front;	Name	Value 🏾 👗
36 SIZE = 0;	this	DList <anytype> (id=26)</anytype>
38	🔺 front	DList\$DNode (id=28)
39⊖ public void add(AnyType elem) {	🔺 elem	null
40	🔺 next	DList\$DNode (id=36)
41 DNode toAdd = new DNode(elem, rear.prev, rear); 42 page prev, payt = toAdd;	prev	null
43 rear.prev = toAdd:	▲ ^F this\$0	DList <anytype> (id=26)</anytype>
44 ++size;	🔺 rear	DList\$DNode (id=30)
45 }	🔺 elem	null
46	🔺 next	null
47 public void add(int index, Anylype elem) throws indexOutOTEC	🔺 prev	DList\$DNode (id=36)
49 if (index < 0 index > size) throw new IndexOutOfBo	▲ ^F this\$0	DList <anytype> (id=26)</anytype>
50	🔺 size	1
51 DNode successor;	0 elem	Integer (id=31)
52 if (index == size) {	📕 value	0
53 successor = rear;	I toAdd	DList\$DNode (id=36)
55 else {	🔺 elem	Integer (id=31)
56 successor = front.next;	∎ ^F value	0
57 int i = 0;	🔺 next	DList\$DNode (id=30)
58 while (i < index) {	▲ prev	DList\$DNode (id=28)
59 successor = successor.next;	▲ ^F this\$0	DList <anytype> (id=26)</anytype>
00 +++1; · · · · · · · · · · · · · · · · · · ·		
< III >	·	P
≫ Debug ⊠	🎉 🕩 BB	🔳 🙌 🚴 👁 . R 🤜 😿 💱 🌄 🗖
80	1	
	N 1	"O 8 🖉



Data Structures & Algorithms

Checking the List Structure

Debugging 24

(×)= Variables 🛛	🕼 🕫 🗖 🎽
Name	Value
this	DList <anytype> (id=26)</anytype>
▲ front	DList\$DNode (id=28)
🔺 elem	null
🔺 next	DList\$DNode (id=36)
▲ prev	null
▲ ^F this\$0	DList <anytype> (id=26)</anytype>
🔺 rear	DList\$DNode (id=30)
🔺 elem	null
🔺 next	null
🔺 prev	DList\$DNode (id=36)
▲ ^F this\$0	DList <anytype> (id=26)</anytype>
🔺 size	1
() elem	Integer (id=31)
🚽 value	0
ItoAdd	DList\$DNode (id=36)
🔺 elem	Integer (id=31)
🖬 value	0
🔺 next	DList\$DNode (id=30)
▲ prev	DList\$DNode (id=28)
▲F this\$0	DList <anytype> (id=26)</anytype>

Well, that looks OK.





Data Structures & Algorithms

Resetting Breakpoints and Resuming

OK, we've confirmed that the first data node is inserted properly; now we can remove the breakpoint at the **for** loop, and set one at the call to the **removeFirstOccurrenceOf()** method, and then click **Resume** to continue execution:



CS@VT

Data Structures & Algorithms

After Resuming... the List is Constructed

Execution proceeds to the new breakpoint:

• 🛛 🖻 🗎 🔅 • O • 💁 • 🤔 🖉 🛷 • 🖗 🌶	a 🗊 🕤 i 🖗					
] ▼ ∰ ▼ \$ \$ \$	• 🗣 🖛	🖹 🕸 Debu	Jav	va
estDriver.java 🛛 🕖 DList.java	- 0	(×)= Variables 🔀		🖄 🍀 🕞	~ - 0)
<pre>public static void main(String[] args) {</pre>	^	Name		Value	*	
SileUniter land		args		String[0] (id=16)		
the f		⊳ Ø log		FileWriter (id=17)		
log - new FileWriter("Log tyt"):		A Q list		Dist Any Types (id=26)		
log - new ritew iter (Log. txt);		a o list		DListCAllyType> (Id=20)		
<pre>DList<integer> list = new DList<integer>():</integer></integer></pre>		A front		DLISTSDINOde (Id=28)		
		👂 🔺 rear		DList\$DNode (id=30)		
for (int $i = 0; i < 10; i++$) {		🔺 size		10		
list.add(i);						
}						
list.display(log);						
<pre>log.write("\n");</pre>						
	=					
list.removeFirstOccurrenceOf(5);						
list.display(log);						
log.write((n);						
list add/E El.						
list display(log):						
log_write("\n"):						
100,111 100((11))						
log.close():						
}						
<pre>catch (IOException e) {</pre>						
System.out.println("IOException: " + e.getMes	sage()):					
1		1				
•		•				
lebug 🛿			🎇 🕩 🔢 🔳	N 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		
			1	e	6 E -	-



Data Structures & Algorithms

Complete List Structure

Debugging 27





Data Structures & Algorithms

Step Into removeFirstOccurrenceOf()

Use **step-into** and proceed to the **while** loop that will walk to the first occurrence of the target value:

111	〗 @ @ 励 棽 ▾ ◙ ▾ 않 ▾ ⑳ @ @ ં⁄? ▾ ♈ ⊿ ☞ []	I II I	御・松・谷・		😰 🏇 Debug 🐉 Ja	ava
J testDri	ver.java 🚺 DList.java 🛿	- 8	(🕬= Variables 🔀		🏝 🍂 🖻 🗸 🗖 🗖	1)
120	AnyType toReturn = target.elem;	•	Name	Value	*	
121	<pre>target.next.prev = target.prev;</pre>		⊿ ⊚ this	DList <anvtype< td=""><td>> (id=26)</td><td></td></anvtype<>	> (id=26)	
122	size:		▶ ▲ front	DListSDNode (i	d=28)	
124	return toReturn:		b A rear	DListSDNode (i	d=30)	
125	}		A size	10		
126	A second second second state of the second		A Q elem	Integer (id=46)		
1279	<pre>public AnyType removeFirstOccurrenceOf(AnyType elem) {</pre>		a value	5		
128	Diada august - front south		A @ current	DList\$DNode (i	d-36)	
129	unde current = tront.next;			Integer (id=21)	u=50)	
131	if (elem.equals(current.elem)) {		A elem	integer (id=51)		
132	AnyType toReturn = current.elem;			0	1.27)	
133	current.prev.next = current.next;		A next	DListSDINode (i	d=3/)	
134	return toReturn;		prev	DListSDNode (i	d=28)	
135	}		▷ ▲ ^r thisS0	DList <anytype< td=""><td>> (id=26)</td><td></td></anytype<>	> (id=26)	
136	current = current.next;					
137	}					
138	return null;					
•		•			P	
The Debug	× ×		8	🛓 🕩 🖩 📕 🛤 🖪 👁 . R	= 😴 😭 ▽ 🗆 🗖	1



In removeFirstOccurrenceOf()

Continue stepping until **current** reaches the node holding the target value:

Debug - DList/src/debug/Examples/DS/DList.java - Eclipse			
<u>File Edit Source Refactor Navigate Search Project Run Window H</u> elp			
1 🕆 - 🗔 🖻 🗟 🏇 - 🔾 - 隆 😂 🔗 - 🍄 🍠 🐲		£] • ∰ • ♥ ♦ • ↔ •	😰 🟂 Debug 🐉 Java
🚺 testDriver.java 🚺 DList.java 🕱	- 6	🗱 Variables 🖾	🖄 🐗 🕞 🎽 🗖 👘
120 AnyType toReturn = target.elem;	^	Name	Value A
121 target.next.prev = target.prev; 122 target prev pext = target pext.		this	DList <anytype> (id=26)</anytype>
123size;		▲ front	DList\$DNode (id=28)
124 return toReturn;		🔺 rear	DList\$DNode (id=30)
125 }		🔺 size	10
126		0 elem	Integer (id=46)
1278 public Anylype removerirstoccurrenceot(Anylype elem) {		🖬 value	5
129 DNode current = front.next;		O current	DListSDNode (id=45)
<pre>plage while (current != rear) {</pre>		🔺 elem	Integer (id=46)
<pre>131 if (elem.equals(current.elem)) {</pre>		∎ ^F value	5
132 AnyType toReturn = current.elem;		🔺 next	DList\$DNode (id=47)
134 return toReturn:	E	▲ prev	DList\$DNode (id=43)
135 }		▲ this\$0	DList <anytype> (id=26)</anytype>
136 current = current.next;			
137 }			
138 return null;	*		
<	•		•
🕸 Debug 🖾		🎉 🕩 🖽	□
			6 7



At End of removeFirstOccurrenceOf()

Continue stepping through the **if** statement and examine the list structure right before the **return** is executed:

testDriver.iava	- 0	(🕬= Variables 😫			
20 AnyType toReturn = target.elem;		Name	Value		
<pre>target.next.prev = target.prev;</pre>		e this	Dist Any Types (id=26)		
<pre>22 target.prev.next = target.next;</pre>		e front	DList(ChiyType> (id=20)	_	
25Size;		a elem	pull	_	
25 }			DLict\$DNode (id=26)	_	
26			Integer (id-21)		
27⊖ public AnyType removeFirstOccurrenceOf(AnyType elem) {		- Fundua		_	
28		a value	Dist\$DNada (id=27)		
29 DNode current = front.next;		A next	Units Divide (Id=37)	E	
if (elem equals(current elem)) {		▲ elem	Integer (Id=38)	=	
AnyType toReturn = current.elem:		e value	1	_	
<pre>current.prev.next = current.next;</pre>		▲ next	DListSDNode (id=39)		
34 return toReturn;		▲ elem	Integer (id=40)		
35 }		e value	2		
<pre>36 current = current.next;</pre>		🔺 next	DList\$DNode (id=41)		
37 }		▲ elem	Integer (id=42)		
39 }		e value	3		
40		🔺 next	DList\$DNode (id=43)		
41⊖ public int size() {		🔺 elem	Integer (id=44)		
42 return this.size;		🖬 value	4		
43 }		🔺 next	DList\$DNode (id=47)		
14		🔺 elem	Integer (id=48)		
16		🖬 value	6		
<pre>47 return (front.next == rear);</pre>		🔺 next	DList\$DNode (id=49)		
48 }		🔺 elem	Integer (id=50)		
49		🖬 value	7		
50⊖ public void clear() {		A next	DList\$DNode (id=51)		
52 front next - rear		A prev	DList\$DNode (id=47)		
53 rear.prev = front:		▲ ^F thisS0	DList <anytype> (id=26)</anytype>		
54 }	-	A prev	DList\$DNode (id=45)	+	-
		4			



Data Structures & Algorithms

List Details

Debugging 31

🕬= Variables 🖾	÷	≫t∎ [□ ▽ □ □]	
Name	Value	· ·	
⊿ ⊚ this	DList <anytype> (id=26)</anytype>		
🔺 🔺 front	DList\$DNode (id=28)		
🔺 elem	null		
🔺 🛕 next	DList\$DNode (id=36)		
🔺 🛕 elem	Integer (id=31)		
e ^F value	0		
a 🔺 next	DList\$DNode (id=37)		
🔺 🔺 elem	Integer (id=38)	-	
e value	1	-	
🔺 🔺 next	DList\$DNode (id=39)		
🔺 🛕 elem	Integer (id=40)		
🖬 value	2		
🔺 🛕 next	DList\$DNode (id=41)		
🔺 🔺 elem	Integer (id=42)		
🚽 value	3		
🔺 🔺 next	DList\$DNode (id=43)		
a 🔺 elem	Integer (id=44)		
🚽 value	4		
a 🛕 next	DList\$DNode (id=47)		
🔺 🔺 elem	Integer (id=48)		
E ^F value	6		
👂 🔺 next	DList\$DNode (id=49)	2	^ ←
📐 🔺 prev	DList\$DNode (id=45)	3	•
bo ▲F this\$0	DList <anytype> (id=26)</anytype>		
b A prev	DList\$DNode (id=41)		
▷ ▲ ^F this\$0	DList <anytype> (id=26)</anytype>		
p a prev	DList\$DNode (id=39)		



Data Structures & Algorithms

Buggered List Structure (more detail)



Apparently the removal method did not correctly reset the **prev** pointer in the node after the node that was removed from the list.

We should check that...



Data Structures & Algorithms

Another Error

A careful exar	nination also reveal	s another bug	
	(X)= Variables 🔀	🏠 📲 🖻 🎽	- 0
	Name	Value	
	I args	String[0] (id=16)	
	⊳	FileWriter (id=18)	
	⊿ [©] list	DList <anytype> (id=22)</anytype>	
	A front	DList\$DNode (id=28)	
	🔈 🔺 rear	DList\$DNode (id=30)	
	🔺 size	10	



A Look at the Code

It should be obvious that two statements are missing from the given code





CS@VT

Data Structures & Algorithms

©2012 McQuain

Debugging 34

Testing Again

Let's execute the modified program:	D:\JavaDevelopment\DList\Log.tx File Edit Search View Encodi
	E Log.txt
	1 0: 0
	² ¹ ; ¹ display of
	⁴ ^{3: 3} initial list
	6 5: 5
	7 6: 6 8 7: 7
	9 8: 8
	10 9: 9
	12 0: 0
	18 6: 7 deleting 5
	19 7: 8
	21
	22 0: 0
Now the list contents seem to be correct. so more	
Now, the list contents seen to be confect so, more	27 5: 5 list after
testing is in order	²⁸ ^{6: 6} ²⁹ ^{7: 7} reinserting 5
	30 8: 8
	31 9: 9 32
	, Normal text file length : 177 lines :



Data Structures & Algorithms

Method Breakpoints

method breakpoint halt when execution enters and/or exits a selected method

To set one, just double-click in the editor margin next to the method header:



By default, this causes a break when execution enters the method...



Breakpoint View

Go to Window/Show View and open the Breakpoint View.

You can right-click on a selected breakpoint to alter its properties:

🌯 Breakpoints 🖾 争 Variables 🗖 🗍				
🗙 💥 🌮 🚭 🔌 🖽 🕞 🔩 🕴 🎓 😘				
VertextDriver [line: 22] - main(String[])		-	Go to File	
DList [entry and exit] - add(AnyType)	-	00	Suspend VM	
			Hit Count	
			Exit	
		\checkmark	Entry	
			Enable	
I Hit count: O Suspend thread Suspend VM V Entry E	xi		Disable	
Conditional Suspend when 'true' Suspend when value changes		×	Remove	
		**	Remove All	
			Select All	Ctrl+A
	\		Сору	Ctrl+C
		Ê	Paste	Ctrl+V
			Import Breakpoints.	
		0	Export Breakpoints	
			Breakpoint Properti	es



Breakpoint View Toolbar



- 1 remove selected breakpoints
- 2 remove all breakpoints
- 3 show breakpoints
- 4 go to file for breakpoint
- 5 skip all breakpoints
- 6 expand all (details)
- 7 collapse all (details)
- 8 link with the Debug View
- 9 set a Java exception breakpoint



Exception Breakpoints

Exception*	
1atching items:	
G AccessControlException - java.security	
G AccessException	
G ^F AccessorException	
G AccountException	
AccountExpiredException	
AccountLockedException	
AccountNotFoundException	
G AclNotFoundException	
ActionNotSupportedException	
ActivateFailedException	
ActivationException	
ActivationSystemException	
Suspend on caught exceptions	
Suspend on <u>u</u> ncaught exceptions	
🖶 java.security - [jre6]	

