Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author’s right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage. http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form.
Enhancements to an Object-Oriented Programming Language

Warwick Bruce Mugridge

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

University of Auckland January 1990
Abstract

The objective of this thesis has been to explore the value and limitations of *Class*, an object-oriented programming language, in order to further the development of the language.

The pivot for this thesis is *WallBrace*, a code-checking system. The development of the *WallBrace* system is the basis of a critique of *Class*, and leads to a number of language extensions being proposed. An important aim in this work has been the careful integration of these enhancements with the rest of the language, avoiding unnecessary additions.

A number of functional and object-oriented extensions to the language are proposed. Discrimination functions, which may be higher-order and polymorphic, add considerable functional power. Generic classes allow for abstract data types, such as sets and lists, to be defined within the language.

The forms interface proposed will greatly enhance the quality of user interfaces to *Class* programs. An external interface will allow *Class* programs to communicate with files, databases, and specialist user-interface programs, such as for plan entry.
I would like to thank John Hosking and John Hamer for many enjoyable discussions over the last five years, in which we have strived for a practical and elegant programming language. Special thanks go to John Hosking, as my supervisor, who has always been most encouraging and supportive.

Thanks are also due to: the staff of the Department of Computer Science for a friendly working environment; Haris Dechapunya at BRANZ ("It all depends on WallBrace!") for his enthusiasm and support; Joe Ten Broeke at BRANZ for providing his knowledge of wall bracing; BRANZ, the University Grants Committee, and the University of Auckland for financial support.

Finally, I would like to thank my parents for their encouragement, and Jackie and Anna for their encouragement, support, and patience while I have been distracted from my life with them.
Contents

1. Introduction
   1.1 Introduction .......................................................... 1
   1.2 Programming Language “Paradigms” ............................. 2
      1.2.1 Functional Languages ........................................... 2
      1.2.2 Object-Oriented Languages .................................... 3
   1.3 Chapter Organisation of the Thesis ................................ 5

2. Introduction to Class
   2.1 Introduction .......................................................... 7
   2.2 Class as a Single-Assignment Language .......................... 7
   2.3 Procedural Aspects of Class ....................................... 8
   2.4 Change and Dependency Management .............................. 9
   2.5 Class as an Object-oriented Language .......................... 10
      2.5.1 Classes .......................................................... 11
      2.5.2 Generalisation Lattice and Inheritance ..................... 15
      2.5.3 Dynamic Classification ........................................ 16
   2.6 Summary ..................................................................... 18
3. **WallBrace**

3.1 Introduction ............................................................................. 21
3.2 The Code ............................................................................... 21
3.3 The Research Project............................................................... 24
  3.3.1 The Problem and the Need for an Application System........ 24
  3.3.2 The Research Project..................................................... 25
3.4 Stage One: Development of a Simple Checker...................... 26
  3.4.1 Object-Centred Representation .................................... 26
  3.4.2 A Goal-Directed Approach ......................................... 28
  3.4.3 User Interface Issues................................................. 29
3.5 Stage Two: A General Checker.............................................. 30
  3.5.1 Wind Loadings on Non-Trivial Buildings....................... 31
  3.5.2 Plan Entry..................................................................... 31
  3.5.3 Roof Structures............................................................. 32
  3.5.4 Calculating Loadings................................................... 33
  3.5.5 Laying Out Bracing Lines & Positioning Diaphragms........ 34
3.6 Class Organisation of WallBrace ............................................ 34
  3.6.1 An Overview................................................................. 34
  3.6.2 Plan Entry..................................................................... 35
  3.6.3 Calculate Loadings....................................................... 36
  3.6.4 Bracing ........................................................................ 36
3.7 Summary ............................................................................. 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Single-Assignment Aspects</td>
<td>39</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Sets and Bags</td>
<td>39</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Need for Functions</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>The User Interface and Procedures</td>
<td>43</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Limitations of a Textual Interface</td>
<td>44</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Single Service: Undue Emphasis on the Report</td>
<td>44</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Coherence of Questions</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>Object-Oriented Aspects</td>
<td>46</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Modularity</td>
<td>47</td>
</tr>
<tr>
<td>4.4.2</td>
<td>The “Type Loss” Problem</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Classification</td>
<td>49</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Multiple Classification</td>
<td>49</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Classification and Inheritance</td>
<td>50</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Sequence: Recursive Classes</td>
<td>51</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Inconsistent Clusters</td>
<td>52</td>
</tr>
<tr>
<td>4.6</td>
<td>Conclusions</td>
<td>54</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Functional Programming Languages</td>
<td>55</td>
</tr>
</tbody>
</table>
5.2.1 Pattern Matching ........................................55
5.2.2 Higher-Order Functions ..................................56
5.2.3 Polymorphic Functions ...................................57
5.2.4 Lazy Constructors .........................................57

5.3 Object-Oriented Languages .................................58
5.3.1 The "Type Loss" Problem .................................58
5.3.2 The "Type Loss" Problem for Functions ...............60
5.3.3 The "Contravariance" Problem .........................62

5.4 Adding Functions to Class .................................64
5.4.1 Simple Functions ........................................64
5.4.2 Functions within Classes ................................65
5.4.3 Higher-Order and Polymorphic Functions ............67
5.4.4 Lazy Functions ...........................................68

5.5 Discrimination Functions .................................69
5.5.1 Simple Discrimination Functions ......................69
5.5.2 Discrimination Functions in Classes ..................70
5.5.3 Narrowing the Result Type ..............................72
5.5.4 A Solution to the Contravariance Problem ...........73
5.5.5 A (Partial) Type-Loss Solution .......................74
5.5.6 Pattern Classes ..........................................75

5.6 Further Type Loss ........................................75
5.6.1 Type Loss ...............................................76
5.6.2 Type Inference ..........................................78
6. **Generic Classes**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>6.2</td>
<td>Functional Languages</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Polymorphic Data Types</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Abstract Data Types</td>
</tr>
<tr>
<td>6.3</td>
<td>Object-Oriented Languages</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Parametric Polymorphism in Classes</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Bounded Parametric Polymorphism</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Contravariance and Generic Classes</td>
</tr>
<tr>
<td>6.4</td>
<td>Adding Generic Classes to Class</td>
</tr>
<tr>
<td>6.4.1</td>
<td>A Simple Example</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Subclasses of a Generic Class</td>
</tr>
<tr>
<td>6.5</td>
<td>An Extended Example: Lazy Lists</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Set/Bag Functions Replaced by List Functions</td>
</tr>
<tr>
<td>6.6</td>
<td>Bounded Parametric Polymorphism</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Sets and Bounded Polymorphism</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Inheritance with Bounded Polymorphism</td>
</tr>
<tr>
<td>6.7</td>
<td>Generators</td>
</tr>
</tbody>
</table>
7. A Forms Interface

7.1 Introduction ............................................. 103
7.2 Functional Languages and I/O.......................... 103
  7.2.1 Streams: Miranda........................................ 104
  7.2.2 Procedures Added: Standard ML...................... 104
  7.2.3 Result Continuations: Hope+C........................ 105
7.3 Providing a Set of Services .............................. 105
  7.3.1 A Program as a Set of Services..................... 105
  7.3.2 Syntel.................................................. 106
  7.3.3 Clausen’s Proposal.................................... 107
7.4 An Extended Forms Interface............................ 108
  7.4.1 Forms and Widgets.................................... 108
  7.4.2 Property Values: Inputs and Outputs............... 110
  7.4.3 Procedures and User Control: Buttons.............. 112
  7.4.4 Classification: Form Elaboration.................. 113
  7.4.5 Components: Subforms................................ 115
  7.4.6 Components and Classification: Tables............... 116
7.5 Applications of the Forms Interface.................... 118
  7.5.1 WallThermalResistance............................... 118
  7.5.2 Hypertext.............................................. 118
7.6 Conclusions .............................................................. 120
7.6.1 Needs of an External Interface .................................. 121
7.6.2 Other Needs of an External Interface .......................... 121

8. External Interface to Class
8.1 Introduction ............................................................. 123
8.2 External Classes ........................................................ 124
  8.2.1 Dependency Issues ............................................... 124
  8.2.2 Properties .......................................................... 125
  8.2.3 Procedures ......................................................... 125
  8.2.4 Using External Classes .......................................... 126
  8.2.5 An Example ....................................................... 127
  8.2.6 Classification ..................................................... 127
  8.2.7 Components ...................................................... 129
8.3 Variable-length Tables ............................................... 134
8.4 Multiple Inheritance and Reference of External Classes .......... 138
8.5 Implementation of the External Interface ......................... 139
  8.5.1 Concurrency and Laziness ...................................... 139
  8.5.2 Connecting to the External Code .............................. 139
  8.5.3 Transferring Data and Control ............................... 140
  8.5.4 Processes in Class ............................................... 140
  8.5.5 Change ............................................................ 141
  8.5.6 Protocol of Interaction .......................................... 142
9. Conclusions and Future Work

9.1 Conclusions ................................................................. 145

9.2 Future Work ............................................................... 146
  9.2.1 Types ......................................................................... 146
  9.2.2 I/O ........................................................................... 147
  9.2.3 Encapsulation ............................................................ 148
  9.2.4 Catalysts for the Further Development of Class .............. 151
Figures

Figure 2.1 A Simple Example .................................................. 8
Figure 2.2 A Simple Example Continued ................................... 9
Figure 2.3 Running a Program .................................................. 10
Figure 2.4 Outline of a Program with Classes .............................. 12
Figure 2.5 Class Structure Diagram for Components ....................... 14
Figure 2.6 Inheritance .......................................................... 14
Figure 2.7 Class Structure Diagram for Generalisation .................... 15
Figure 2.8 Conformance .......................................................... 15
Figure 2.9 Multiple Inheritance ............................................... 16
Figure 2.10 Classification: BracingElement Extended ...................... 17
Figure 2.11 Classification ....................................................... 17
Figure 2.12 Boolean Classification ........................................... 18

Figure 3.1 Wind Loads on a Building ......................................... 22
Figure 3.2 Clause 6.3.2.1 of NZS 3604 ..................................... 27
Figure 3.3 An Example Building Plan ......................................... 31
Figure 3.4 Roof Overlaps ...................................................... 33
Figure 3.5 Clause 6.3.5.2 of NZS 3604 ..................................... 33
Figure 3.6 Clause 6.9.6.3 of NZS 3604 ..................................... 33
Figure 3.7 Overview of Classes in WallBrace ........................................35
Figure 3.8 Varied-Storey Building Plan .............................................36

Figure 4.1 Use of the Function any ..................................................40
Figure 4.2 The Class Rectangle .......................................................40
Figure 4.3 Other Use of Sets ..........................................................41
Figure 4.4 Repetitious Code ............................................................42
Figure 4.5 Class Range .................................................................42
Figure 4.6 Using Classes as Functions .............................................43
Figure 4.7 An Example of Interface Control .................................46
Figure 4.8 Wall Class Family ..........................................................48
Figure 4.9 Classification Type Information is "Lost" .........................49
Figure 4.10 Classification in Roof .................................................49
Figure 4.11 Multiple Classification ...............................................50
Figure 4.12 Without Classification .................................................50
Figure 4.13 Generalisation in the Roof Class Family ......................51
Figure 4.14 Recursive Wall Class Family ........................................52
Figure 4.15 A Recursively-Defined Sequence ................................52
Figure 4.16 Inconsistency with Classification ................................53
Figure 4.17 Invalid Multiple Inheritance ........................................53
Figure 4.18 Mutually-Exclusive Classes .......................................54
Figure 5.1 Pattern Matching .................................................. 56
Figure 5.2 Higher-Order Function ......................................... 56
Figure 5.3 Fold: A Polymorphic Function .............................. 57
Figure 5.4 Using Fold ......................................................... 57
Figure 5.5 Infinite Structures through Laziness ...................... 58
Figure 5.6 Incorrect Code .................................................... 59
Figure 5.7 Type_case in Trellis/Owl ...................................... 59
Figure 5.8 Multi-Methods in CLOS ........................................ 60
Figure 5.9 Type Loss Problem .............................................. 60
Figure 5.10 Avoiding “Type Loss” in Eiffel ............................ 61
Figure 5.11 Avoiding “Parameter Loss” ................................. 62
Figure 5.12 Contravariance Problem in Eiffel ....................... 63
Figure 5.13 Separate Interfaces ............................................. 64
Figure 5.14 An Example Function ......................................... 65
Figure 5.15 Public Functions ................................................ 66
Figure 5.16 Using a Public Function ....................................... 66
Figure 5.17 IntList Class Family ......................................... 67
Figure 5.18 Using filter, map, and fold ................................. 68
Figure 5.19 Use of Lazy Lists .............................................. 68
Figure 5.20 Strict Form of Function fromTo ........................... 69
Figure 5.21 Classification Type Information Regained .............. 70
Figure 5.22 Discrimination Function for Function last .............. 71
Figure 5.23 A Discrimination Function .................................. 71
Figure 5.24 Using addList.................................................71
Figure 5.25 A Discrimination Function..............................72
Figure 5.26 Contravariance and Parameter-Loss Solution........73
Figure 5.27 Consistency of result from equal.........................74
Figure 5.28 Defining firmEqual Outside the Classes.................74
Figure 5.29 Using Function move.....................................74
Figure 5.30 Pattern Matching and Classification....................75
Figure 5.31 Revised IntList Class Family............................76
Figure 5.32 Inheriting filter in the ExtIntList Classes............77
Figure 5.33 The Relationship between the Classes..................77
Figure 5.34 Mixing a Type with a Super Type........................82
Figure 5.35 Introducing another Family..............................83
Figure 5.36 Mixing Types.............................................83
Figure 5.37 A Problem with “like Current”...........................84

Figure 6.1 User-Defined Data Types................................88
Figure 6.2 An Abstract Data Type (Standard ML)..................88
Figure 6.3 Generic Class SET.........................................89
Figure 6.4 Using the Generic Class SET..............................90
Figure 6.5 Bounded Type Variables..................................90
Figure 6.6 Constraining the Type....................................90
Figure 6.7 Generic Class Example.................................91
Figure 6.8 Generic Parameters in a Subclass.........................92
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9</td>
<td>Generic Class Stack and its Subclass</td>
<td>92</td>
</tr>
<tr>
<td>6.10</td>
<td>An Integer Stack</td>
<td>93</td>
</tr>
<tr>
<td>6.11</td>
<td>Lazy Lists</td>
<td>94</td>
</tr>
<tr>
<td>6.12</td>
<td>Collect and Select</td>
<td>95</td>
</tr>
<tr>
<td>6.13</td>
<td>Collect with Map and Select with Filter</td>
<td>95</td>
</tr>
<tr>
<td>6.14</td>
<td>Filtering Subclasses</td>
<td>95</td>
</tr>
<tr>
<td>6.15</td>
<td>Create from Map</td>
<td>96</td>
</tr>
<tr>
<td>6.16</td>
<td>Set Family</td>
<td>97</td>
</tr>
<tr>
<td>6.17</td>
<td>Class Comparable</td>
<td>97</td>
</tr>
<tr>
<td>6.18</td>
<td>Invalid Inheritance</td>
<td>98</td>
</tr>
<tr>
<td>6.19</td>
<td>Valid Inheritance with a Discrimination Function</td>
<td>98</td>
</tr>
<tr>
<td>6.20</td>
<td>The AskGenerator Class Family</td>
<td>99</td>
</tr>
<tr>
<td>6.21</td>
<td>An Example of the use of AskGenerator</td>
<td>100</td>
</tr>
<tr>
<td>6.22</td>
<td>The Class Relationship</td>
<td>100</td>
</tr>
<tr>
<td>6.23</td>
<td>The Generator Family</td>
<td>101</td>
</tr>
<tr>
<td>6.24</td>
<td>AskGenerator2 With Generator</td>
<td>101</td>
</tr>
<tr>
<td>6.25</td>
<td>Class Tree Written in a Functional Style</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Synchronising Streams in Miranda</td>
<td>104</td>
</tr>
<tr>
<td>7.2</td>
<td>I/O in Standard ML</td>
<td>104</td>
</tr>
<tr>
<td>7.3</td>
<td>An Example Form</td>
<td>109</td>
</tr>
<tr>
<td>7.4</td>
<td>Multiple Forms on the Screen</td>
<td>111</td>
</tr>
<tr>
<td>7.5</td>
<td>An “Error” Property</td>
<td>111</td>
</tr>
</tbody>
</table>
Figure 9.2 Module for ListFamily ................................................................. 149
Figure 9.3 Sub-Families of List ................................................................. 150
Figure 9.4 The GeneratorFamily and a Sub-Family .............................. 151