Index

-A-

Accessibility:
of data members, 173
of member functions, 179–180
Accessor functions, 177
implementation of, 195
naming of, 177
return type of, 182
signature of, 182
Adams, Sam, 29
Analysis:
and CRC cards, 30–31
common vocabulary, 83
holes in requirements, 84
live prototyping, 83
mind shift, 83
spreading knowledge, 83
and design, 82, 103
in software development, 20
Annotone, Dave, 65
Assignment operator:
implementation of, 194–195

[Assignment operator:]
base class member assignment, 194
check for assign to self, 194
need for, 176
return type of, 182
signature of, 182
Association, in OMT, 99
Attribute descriptions, in Shlaer/Mellor, 98
Attributes:
in C++, 159
on CRC card, 34
in CRC card session, 70
for design, 112
identification of, 54–55
in isolation, 55

-B-

Base class, 160
abstract, 161
concrete, 161
Baseball, object-oriented model of, 22–24
Beck, Kent, 28, 57
Booch methodology:
  and CRC cards, 95
  class diagram, 94–95
  clouds, 94–95
  object diagram, 94–95
  scripts, 97
Booch, Grady:
  methodology, 99
  quote, 106
Brooks, Ned, 69, 133
Budd, Timothy, quote, 149, 150

-C-
CardCase, 29
CCOOL, 139
Class:
  in C++, 152–153 (see also C++ class)
  on CRC card, 32–33
  defined, 12
  documentation of, 143–144
  interface, 89
  vs. object, 11–12
Class descriptions:
  on CRC card, 34, 52
  in RDD, 93
Class diagram:
  in Booch methodology, 94–95
  in OMT, 99
Class libraries, 109–110
Class, Responsibility, Collaborator (see CRC)
Classes:
  for analysis, 81, 84–85
  brainstorming, 49
  for design, 111
  filtering, 50
  finding, 49

Clouds, in Booch methodology, 94–95
C++:
  ANSI standardization of, 150
  history of, 150
  and object-oriented concepts, 21
  and polymorphism, 17
  relationship to C, 150
C++ class:
  from CRC card, 163–164
  base, 160–161
  abstract, 161
  concrete, 161
  derived, 160
  nested, 158
  parameterized, 161, 162
  template, 162
  wrapper, 90
Coad/Yourdon, 29
Collaborating responsibility:
  in C++, 162
  on CRC card, 114–115
Collaboration drawings, in session, 72–74
Collaboration graphs, in RDD, 93
Collaboration:
  in C++, 154–159
    has-a relationship, 155–157
    uses-a relationship, 157–160
    direction of, 33, 59
    documentation of, 142–143
    and has-a relationship, 61
Collaborator:
  in C++, 154–159
  on CRC card, 33
  defined, 11
Collection classes, 119
Const:
  cast away, 186
  member function specifier, 185–187
INDEX

Constructor:
    declaration of, 174
    implementation of, 191–193
    initialization list, 192
    return type of, 181
    signature of, 181
Coplien, Jim, 29
Copy constructor:
    declared private, 176
    need for, 175
    return type of, 182
    signature of, 182
CRC, 1.28
    mapping to C++, 152–163
CRC card:
    attributes, 34
    back of, 34
    class, 32–33
    collaborator, 33
    defined, 27
    responsibility, 33
    size of, 32, 34–35
    subclass, 33
    superclass, 33
    syntax of, 32–34
CRC card session, 41–78
    choosing alternatives, 61
    facilitating, 77–78
    facilitator, 44, 50
    group dynamics of, 45
    participants, 43–45, 82
    potential pitfalls, 75–77
      deadlock, 59, 75–76
      getting lost, 76–77
    preparation for, 42–47
    role of managers in, 45
    scribe, 57
    size of group, 42–43
CRC card tools:
    CCOOL, 139
    CRCCards, 139–140
    Object System/CRC, 138–139
    role of, 28, 137
CRC cards:
    and Booch methodology, 95
    and OMT, 100–101
    and OOSE, 97
    and RDD, 94
    and Shlaer/Mellor, 98–99
    advantages, 34–36
    assignment of, 52, 64
    flexibility of, 37–38
    grouping of, 71–72
    history of, 28–29
    and methodologies, 29, 91–101
    and object-oriented requirements, 29
    and organizational modeling, 29
    and patterns, 29
    session (see CRC card session)
    in software development, 29–31
    strengths in analysis, 82–84
    strengths in design, 104–106
    tools, 136–141
CRC Cards, 139–140
Cunningham, Ward, 28

-D-

Data members:
    choice of, 164–173
    of object type, 166–169
    of pointer type, 165–168
    private, 173
    of reference type, 165
    static, 172–173
    types of, 165–168
<table>
<thead>
<tr>
<th><strong>Database design</strong></th>
<th>113</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database management system</strong></td>
<td>(see DBMS)</td>
</tr>
<tr>
<td><strong>DBMS</strong></td>
<td>108–109</td>
</tr>
<tr>
<td>object-oriented</td>
<td>109</td>
</tr>
<tr>
<td>relational</td>
<td>109</td>
</tr>
<tr>
<td><strong>Decomposition:</strong></td>
<td></td>
</tr>
<tr>
<td>algorithmic</td>
<td>13–15</td>
</tr>
<tr>
<td>algorithmic vs. object-oriented</td>
<td>15–16</td>
</tr>
<tr>
<td>object-oriented</td>
<td>2, 15–16</td>
</tr>
<tr>
<td><strong>Derived class</strong></td>
<td>160</td>
</tr>
<tr>
<td><strong>Design:</strong></td>
<td></td>
</tr>
<tr>
<td>and CRC cards</td>
<td>31</td>
</tr>
<tr>
<td>framework for code</td>
<td>105</td>
</tr>
<tr>
<td>informal notation</td>
<td>105</td>
</tr>
<tr>
<td>for reviews</td>
<td>105</td>
</tr>
<tr>
<td>spread of expertise</td>
<td>105</td>
</tr>
<tr>
<td>and analysis</td>
<td>82, 103</td>
</tr>
<tr>
<td>and implementation</td>
<td>104</td>
</tr>
<tr>
<td>in software development</td>
<td>20</td>
</tr>
<tr>
<td><strong>Design constraints:</strong></td>
<td></td>
</tr>
<tr>
<td>environment</td>
<td>106</td>
</tr>
<tr>
<td>language</td>
<td>107</td>
</tr>
<tr>
<td>memory</td>
<td>110</td>
</tr>
<tr>
<td>performance</td>
<td>110</td>
</tr>
<tr>
<td>security</td>
<td>110</td>
</tr>
<tr>
<td>software components</td>
<td>107–110</td>
</tr>
<tr>
<td><strong>Destructor:</strong></td>
<td></td>
</tr>
<tr>
<td>declaration of</td>
<td>174</td>
</tr>
<tr>
<td>implementation of</td>
<td>193</td>
</tr>
<tr>
<td>destroy responsibility</td>
<td>193</td>
</tr>
<tr>
<td>return type of</td>
<td>181</td>
</tr>
<tr>
<td>signature of</td>
<td>181</td>
</tr>
<tr>
<td>virtual</td>
<td>174, 189</td>
</tr>
<tr>
<td><strong>Documentation:</strong></td>
<td></td>
</tr>
<tr>
<td>of CRC card design</td>
<td>141–147</td>
</tr>
<tr>
<td>class descriptions</td>
<td>143–144</td>
</tr>
<tr>
<td>class model</td>
<td>142–143</td>
</tr>
</tbody>
</table>

[Documentation:] |
| scenario descriptions | 145–147 |
| UNIX manual page | 143 |
| with formal methods | 91–101 |
| **Domain expert** | 44 |
| **Domain idiot** | 78 |
| **Dynamic model, in OMT** | 99 |

**E** |

Encapsulation | 9 |
**Enhancements:** | |
in C++ | 162 |
to CRC cards | 113–116 |
**Existing systems:** | |
and CRC cards | 88–90 |
full-scale approach for | 88 |
for one subsystem | 89 |
reengineering of | 90 |
**Extensibility, of object-oriented model** | 17 |

**F** |

Facilitator | 44, 50, 77–78 |
**File organization:** | |
in C++ | 197–199 |
forward declarations | 199 |
header files | 198–199 |
preprocessor guards | 198 |
source files | 199 |
Flanagan, Liz | 66, 126 |
**Formal methodologies** | (see Methodologies) |
**Frameworks:** | |
of domain classes | 91 |
user interface | 108 |
Friend function | 183 |
Functional model, in OMT | 99 |
Index

-G-
Glor, Bronwyn, 63
Grafton, Sue, quote, 27
Graphical user interface (see GUI)
Group:
dynamics of, 45
make-up of, 43–45
size of, 42–43
GUI, 108

-H-
Has-a relationship:
in C++, 157–159
in modeling, 71
Hayes, Katie, 56
Hierarchy graphs, in RDD, 93

-I-
Implementation:
and design, 104
of member functions, 190–197
in software development, 20
Incremental development, 20
Index cards, advantages, 28, 36–37
Information hiding, 9
Information model, in Shlaer/Mellor, 97
Inheritance, in C++, 160
Initialization list, for constructor, 192
Inline function:
outlined, 187
specifier, 187
Instance diagram, in OMT, 99
Interaction diagram:
in OOSE, 96
for scenario documentation, 145
Interface class, 89, 117–119

-J-
Jacobson, Ivar, inventor of OOSE, 96

-L-
Lippman, Stan, 151

-M-
Main routine, in C++ program, 200
Manager classes, 117
Member functions:
accessibility of, 179
const, 185–187
implementation of, 190–197
inline, 187
from responsibilities, 174–179
pure virtual, 161, 190
signature of, 180–185
specifiers, 185–190
virtual, 187–191
Memory management, in C++ class, 166–167
Message, sending of, 11
Message data:
in C++, 162
on CRC card, 115–116
Methodologies:
and CRC cards, 91–101
data-driven, 92
kinds of, 92
responsibility driven, 92
Methodology, defined, 4
Meyers, Scott, 151, 162
Mind shift, 8
[Mind shift:]
  using CRC cards, 30
Murray, Rob, 151
Mutator functions, 177
  implementation of, 195
  naming of, 177
  return type of, 182
  signature of, 182

-N-

Naming conventions, for C++, 151
Nested class, 158

-O-

Object:
  behavior of, 10
  defined, 9
  identity of, 10–11
  interface of, 10
  operations of, 10
  state of, 9–10
Object Communication Model, in Shlaer/Mellor, 99
Object Modeling Technique (see OMT)
Object System/CRC, 138–139
Object descriptions, in Shlaer/Mellor, 98
Object diagram, in Booch methodology, 94–95
Object model, in OMT, 99
Object-Oriented Software Engineering (see OOSE)
Object-oriented development, 20
  defined, 4
  history of, 91–92
Object-oriented model, 4, 12
Object-oriented programming, 7, 21

OMT, 99
  and CRC cards, I00–I01
  association, 99
  class diagram, 99
  dynamic model, 99
  functional model, 99
  instance diagram, 99
  object model, 99
OOSE, 96
  and CRC cards, 97
  interaction diagram, 96
  use case, 96
  use case hierarchy, 97

-P-

Parameterized class, 161–162
Pittman, Matthew, 78
Polymorphism:
  in C++, 187
  in modeling, 17
Problem model:
  for Library application, 87
  stabilizing, 86–87
Process model:
  in Shlaer/Mellor, 98
Programming:
  object-oriented, 7, 21
  structured, 7
  structured vs. object-oriented, I6
Prototyping, 20
  when to begin, 135–136

-R-

RDD, 29,93
  and CRC cards, 94
  class descriptions, 93
INDEX

[RDD:] collaboration graphs, 93
    hierarchy graphs, 93
    and subsystems, 93
Relationship descriptions, in Shlaer/Mellor, 98
Responsibilities:
    for analysis, 85–86
derived from requirements, 54
    for design, 112
    identification of, 54
    in isolation, 54
    and member functions, 174–179
Responsibility:
in C++, 153–154
    on CRC card, 33
defined, 10
Responsibility Driven Design (see RDD)
Reuse, 90–91
Rogue Wave Library, 110, 118
Ross, Stephanie, 56
Rumbaugh, James, 99

-S-
Scenario list (see scenarios, list of)
    for analysis, 85
    check out, 57–62, 122–126
    for creation, 120–121
defined, 18, 55
    for design, 112
documentation of, 145–147
    encompassing, 56
    end-to-end, 56
    exceptional, 66–67
    execution of, 4
guidelines for execution, 57
    for implementation, 163

[Scenario list:]
    inside, 56
    list of, 72
    ordering of, 56
    relationships among, 145–146
    return, 66, 126–133
    search, 69–71, 133–135
    specificity of, 56, 134
    for system start-up, 122
    as tests, 163
Scripts, in Booch methodology, 97
Session (see CRC card session)
Shlaer/Mellor:
    attribute descriptions, 98
    and CRC cards, 98–99
    information model, 97
    object descriptions, 98
    OCM, 99
    process model, 98
    relationship descriptions, 98
    state model, 98
Software Development:
    object-oriented, 20
    history of, 91–92
    waterfall model, 20
Software components:
    class libraries, 109
    for data management, 108–109
    for user interfaces, 108
Specifiers:
    const, 185–187
    inline, 187
    public, 161
    virtual, 187–191
State of, 9–10
Stewart, Barbara, 57, 123
Stroustrup, Bjarne:
    and C++, 150, 151
226

[Stroustrup, Bjarne:]
quote, 46
Structured programming, 7
Subclass:
on CRC card, 33
defined, 16
Subclass:
in session, 52–53, 63
Subresponsibilities:
in C++, 154
on CRC card, 76, 113–114
Subsystems:
in CRC card session, 47, 72
in RDD, 93
Superclass:
abstract, 16
concrete, 17
on CRC card, 33
defined, 16
in session, 52–53, 63
Superclass/subclass relationship:
in C++, 160
defined, 16
documentation of, 142
in session, 52–53, 63

-T-

Template classes, 161–162
Thomas, Dave, 29

-U-

Use case, in OOSE, 96
Uses-a relationship:
in C++, 155–156
in design, 155–156

-V-

Virtual function:
pure, 161, 190
specifier, 187–191

-W-

Waterfall model, 20
Wirfs-Brock, Rebecca:
attribute wisdom, 51
inventor of RDD, 29, 93
quote, 103
Wrapper class, 90

-Z-

Ziegler, Chris, 56
Ziegler, Garrett, 67