The Open University

Mathematics and Computing: Level 2
M255 Object-oriented programming with Java

M255 Index

Copyright © 2006 The Open University
1.1

WEB 86855 1

The references for the entries in this index are given by the unit number as an emboldened numeral followed by the page number. For example, **1**.19 refers to *Unit 1*, page 19.

A
abstract
  class **6**.39
  method **6**.39
abstraction **14**.10
acceptance testing **13**.14
access modifiers **6**.17
accessor
  message **2**.23
  methods **4**.26, 28
activation rectangle **14**.23
actual results **13**.12
adaptive **13**.11, **14**.11
adaptive method **14**.33
agile development process **14**.35
alpha testing **13**.13
analysis **14**.12
analysis pattern **13**.31, **14**.31
and **5**.34
application **1**.10
application domain **1**.18
argument **1**.21, **2**.17, **4**.38
array **9**.5, 7
`ArrayList` **11**.7
ASCII **3**.7
assertion **13**.16
assignment statement **3**.9
attributes **1**.19
attribute value **1**.19
autoboxing **10**.13

B
behaviour **1**.19, **2**.8
  state-dependent **2**.14
beta testing **13**.13
binary digit **1**.7
binary operator **3**.21, **5**.33
bit **1**.7
black box testing **13**.6
BlueJ **3**.9
`boolean` **3**.8
Boolean
  condition **5**.21, 24
  expression **5**.33
  operators **5**.34
boundary error **13**.47
buffer **9**.63, **12**.24
bug **8**.32
`byte` **3**.7
bytecode **1**.15, **4**.19

C
call stack **8**.35
Cancel button **5**.15, 31
capacity **9**.63
CASE (computer-aided software engineering) tools **14**.39
cast **3**.38
catch **8**.34
chained **7**.42
`char` **3**.7
checked exception **8**.41
class **1**.44, **2**.8
  diagram **14**.17
  header **6**.11
  method **5**.7, **7**.37
  variables **7**.35, 48
classification **6**.6–7
client **12**.56, **14**.10
code **1**.7
code-based testing tool **14**.40
coding tool **14**.39
collaborating objects **2**.23, 26
collection **9**.5, 8,
collection classes **10**.6, **11**.5, 7, 23
  terminology **10**.36
`Collections` **11**.19
comma-delimited file (CSV) **12**.37
comment **4**.8
common message protocol **6**.39
`Comparable` **11**.26
`compareTo()` **11**.26
comparison operators **5**.33
compiler **1**.14, **4**.5
compile-time errors **8**.27
component **1**.47, **9**.7
  type **9**.9, 42
Component-Based Development (CBD) **1**.30
compound expression **3**.25
computation error **13**.47
computer system **1**.8
concatenate **3**.32
concatenation operator **5**.17
concrete class **6**.38
condition **5**.21
conditional selection **5**.5
`confirm` **5**.12
constant **7**.44
constant instance variable **7**.45
constructor **3**.14, **6**.19
constructor chaining **6**.20

D
data field **6**.9, **13**.6
data hiding **4**.52
debugging **4**.13, **8**.42, **13**.6
decimal number **3**.7
declaration **3**.26
delimiter **3**.32
delta **13**.58

design **14**.12
- designing dynamic models **14**.11
- detailed design and implementation **14**.11
- principle **7**.17, 30
- tool **14**.39

designer **14**.38

destructive **10**.38, **11**.19

development **14**.11

dialogue boxes **5**.6

difference **10**.38

direct
- access to instance variables **7**.47
- interaction **7**.9–10
- subclass **6**.10
- superclass **6**.10

domain **1**.18

domain model **14**.8

`double` **3**.7

dynamic
- compilation **1**.16
- model **14**.24
- semantic error **8**.34

E

editor **4**.5

effective length **9**.15

element **9**.8

element type **9**.10, 42, **10**.7

encapsulation **4**.52

entry **10**.23

equal to **5**.33

equality operators **3**.23, **5**.33

`equals()` **11**.34

equivalence class **13**.48

escape character **12**.9

evaluate **3**.21

exception **8**.34
- checked **12**.16
- handler **8**.35
- handling **8**.34
- unchecked **12**.15

expected results **13**.12

expression **3**.21

extends **6**.11, 21

eXtreme Programming (XP) **14**.35

F

`false` **3**.8

fields **6**.9

`File` **12**.9

`final` **7**.44, **9**.55

finally **12**.28

fixed-size **9**.7, 32

`float` **3**.7

floating-point number **3**.7

floating-point types **3**.7

`for` loop **5**.44

`foreach` statement **9**.26

formal argument **4**.39

formatting guidelines **8**.6

G

garbage collection **3**.17

general-purpose classes **7**.51

getter
- message **2**.23
- methods **4**.26

greater than **5**.33

greater than or equal to **5**.33

H

hash bucket **11**.37

hash code **11**.37

`hashCode()` **11**.37

`HashMap` **10**.22

`HashSet` **10**.7

helper method **6**.18, **7**.17

hiding **7**.38

high-level language **1**.13

homogeneous **9**.7

I

identifier **3**.18, **14**.20

identity **9**.60

`if-then` statement **5**.21

`if-then-else` statement **5**.21

immutable **9**.62

implementation **14**.12

`implements` **6**.48

increment **13**.14

index **9**.7

indexable **9**.7

indirect
- interaction 10
- subclass **6**.10
- superclass **6**.10

inherit **6**.7

initialisation **2**.8
- of class variables **7**.36

initialise **2**.8

input box **5**.14–15

`InputStream` **12**.7

instance **1**.44, **2**.6
- protocol **1**.45
- variables **3**.17, **4**.21, 28

`instanceof` **12**.49

instantiate **6**.40

`int` **3**.7

`Integer` **5**.18

integer
- division **3**.21
- types **3**.6

integrated development environment 5

integration testing **13**.14

interface **6**.47
- type **10**.9

intermediate code **1**.15

interpreter **1**.14–15

intersection **10**.38

iteration **5**.5, **11**.12, **14**.33

iterative method **14**.33

J

.jar file **9**.56

Java Class Libraries **7**.51

Java Collections Framework **10**.7, **11**.5, 17

Javadoc **4**.17, **6**.8

`java.io` **12**.7–8

`java.lang` **7**.51

`java.util` **10**.9

`java.util.Arrays` **9**.46

Java Virtual Machine **1**.16

JUnit **14**.40

just-in-time compilation **1**.16

K

key **10**.20

key–value pair **10**.21

keyword **3**.18

L

large-scale project **14**.36

lazy evaluation **5**.40

length **9**.10

less than **5**.33

less than or equal to **5**.33

lifeline **14**.23

link **14**.18, 20

list **11**.6

literal **3**.7

local variable **5**.12, **6**.17, **7**.46

logical error **8**.32

logical operators **3**.23

`long` **3**.7

low-level **1**.8

low-level language **1**.13

M

`main()` **9**.49

maintainability **7**.24

maintenance **14**.11

manifest file **9**.56

map **10**.20

mapping **10**.23

masking **7**.38

member **7**.39

message
- accessor **1**.18, **2**.23
- answer **2**.23
- expression **3**.30
- get **2**.23
- name **2**.18
- set **2**.23

message-send **1**.44, **2**.32

method **4**.5, 8
- body **4**.8
- header **4**.8, **6**.14
- invocation **4**.19
- signature **4**.39

Method Summary **6**.9

microworld **1**.38

modal dialogue box **5**.6

model **1**.18

modelling language **14**.14

N

native methods **7**.7

natural ordering **11**.5, 19

nested expression **3**.25

`new` **3**.14, **6**.19

non-destructive **10**.38

not **5**.34

not equal to **5**.33

`null` **3**.14, **5**.15, **6**.20

number types **3**.6

O

object **1**.19
- collaboration **2**.23, 26
- creation **2**.29
- diagram **14**.19

`Object` **6**.10

Object Bench **13**.21

Object Management Group (OMG) **14**.15

Observer pattern **13**.55

object-state diagram **2**.27

operating system **1**.8

operator **3**.21

or **5**.34

orchestrating instance **7**.19

ordered collection **11**.19

`OUDialog` **5**.6

`OUFileChooser` **12**.10

`OutputStream` **12**.7

OUWorkspace **3**.9

overloading **6**.24, **7**.13, 15

overload-resolution **7**.14, **7**.15

overriding **6**.11, **7**.15

P

parentheses **5**.33

parse **8**.27

`parseInt()` **5**.18

pattern **6**.28, 32

peripheral device **1**.7

persistence **12**.5, 33, 36

phase **14**.10

polymorphic
- method **6**.45, 52
- variable **6**.45, 52

polymorphism **2**.10

port numbers **12**.56

precedence rules **3**.24

predictive method **14**.33

primary sort **11**.31

primitive data type **3**.6, **4**.35

`private` **4**.22, 25, **6**.18

problem domain **1**.18, **14**.22, 37

procedural programming **1**.17

program **1**.9

programmer **14**.38

project
- documentation **14**.38
- failure **14**.36
- management **14**.36
- manager **14**.38

`protected` **6**.18, **7**.48

protocol **1**.45

prototype **14**.34

pseudo-variable **4**.10, 23

`public` **4**.25, **6**.18

Q

qualified **7**.37, 48

R

readability **5**.14

`Reader` **12**.7, 21

receiver **1**.44

recursion **6**.14

recursive method **6**.14

refactoring **6**.40, **7**.25

reference semantics **3**.20

reference type variable **3**.13

regression testing **13**.14

relational operators **3**.23, **5**.33

requirements **14**.5
- analyst **14**.37
- specification **14**.11

responsibility **2**.35

review **14**.34

risk analysis **14**.36

run-time
- error **8**.32
- system **1**.14

S

`Scanner` **12**.38

scope **8**.21

secondary sort **11**.31

semantic error **8**.28

sequence diagram **2**.24, 35, **14**.23

serialisation **12**.5, 50

server **12**.56

set **10**.6

setter
- message **2**.23
- methods **4**.25

`short` **3**.7

short-circuit evaluation **5**.40

`size()` **10**.10, **11**.7

socket **12**.56

software **1**.7
- components **1**.47
- developer **14**.37
- development **14**.6
- development method **14**.31, 36
- engineering **14**.36
- model **14**.13
- quality **14**.36
- system **1**.8
- tester **14**.38

source code **1**.7

stable sort **11**.31

standard default value **6**.20

state **1**.20

state (of software) **14**.21

state-dependent behaviour **2**.14

statement **3**.8–9

statement block **5**.22

`static` **5**.7, **7**.36

static
- constants **7**.52
- model **14**.24
- variables **7**.48

stream **12**.6

strict UML diagrams **14**.15

string **9**.57

`StringBuilder` **9**.63

structural model development **14**.11

sub-array **9**.31

subclass **2**.14, **6**.7, 10

sub-expression **3**.25

subinterface **10**.33

sublist **11**.20

substitutability **6**.45, 52

substitution **6**.52

`super` **6**.19

superclass **2**.14

superinterface **10**.33

syntax **3**.20
- error **8**.27

system **1**.8
- software **1**.8
- testing **13**.14

`System.out` **7**.51

systems analyst **14**.37

T

technical writer **14**.38

test data **13**.12

testing **13**.6, **14**.11
- framework **13**.15
- techniques **14**.36
- tool **14**.40

test driver tool **14**.40

test-driven development (TDD) **13**.14

test-first approach **13**.14

test fixture **13**.30

test harness **13**.14

`this` **4**.10, **6**.12

throw **8**.34

token **12**.37

`TreeMap` **11**.32

`TreeSet` **11**.18, 32

`try–catch` statement **8**.38, **12**.16

`true` **3**.8

two-dimensional array **9**.38

type **4**.35

U

UML (Unified Modeling Language) **14**.14

UML standard **14**.15

UML-type diagrams **14**.15

unary operator **3**.22, **5**.33

unchecked exception **8**.41

Unicode **3**.8

union **10**.38

unit testing **13**.5, 13

user interface development **14**.11

V

`valueOf()` **5**.18

value semantics **3**.20

value type variable **3**.9

variable **1**.39, **2**.27, **3**.6

variable (different types) **7**.46

variable reference diagram **3**.13
  different sorts of **7**.46

virtual machine **1**.19

visibility **7**.48

visual representation **1**.37

W

waterfall method **14**.32

`while` loop **5**.52

white box testing **13**.7

workspace variable **7**.46

wrapper class **10**.13

`Writer` **12**.7, 17