Introduction to Java — PL/SQL Developers Take Heart!

Peter Koletzke
Technical Director &
Principal Instructor
ORACLE ACE Director


quovera

utoug utah oracle users group

Dennis Should Know

A language that doesn't affect the way you think about programming is not worth knowing.

—Dennis M. Ritchie

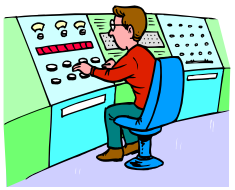
Survey

- Years with PL/SQL?
  - Less than 2, 2-4, 4+
- Years with Java?
  - None
  - 1-2, 2-4, 4-15, 16+
- Other languages?
  - C and/or C++
  - Smalltalk
  - FORTRAN, COBOL, Basic, JCL, Perl, REXX …

Agenda

- Foundations
- Parts list
- Language basics

Slides and white paper will be on the UTOUG and Quovera websites.



Why and What

- Platform independent (the promise of portability)
- The core of Java EE (formerly "J2EE")
- A well-developed user community
- Highly-evolved language – v8

  - Competitor to .NET (Java EE 7)
  - Oracle owns Java
- Looks like C++, but …
  - No pointers; manages memory for you (C-- ?)
- Very different from PL/SQL
  - Needs an object-oriented thought process

Big Three OO Concepts

- Inheritance
  - Parent-child relationship for classes
  - Child (subclass) has data and behavior of the parent
  - Nothing like this in PL/SQL
- Encapsulation
  - Data (in variables) is not accessible directly
  - You must use an approved interface to get to data
    - `setCity()`, `getAddress()`, etc.
  - Like package functions and procedures that set and retrieve package body variables
- Polymorphism
  - Caller doesn't know which method will be called
  - Similar to overloading in PL/SQL
- Java and PL/SQL use different paradigms

OO Basics

- Class
  - The fundamental building block
  - A pattern, template, archetype, or blueprint from which objects are built
    - Like for a car – 2007 Toyota Prius
  - A "concept" not anything "real"
  - Also called an *abstract data type*
- Objects
  - "Real" things – in code, anyway
    - Like PL/SQL variables
  - The "instantiation" of a class
    - 2007 Toyota Prius (VIN 785789359019)
  - Kind of like a variable built from a data type

Another Way to Think About Objects

- Kind of like PL/SQL variables
  - Each variable (instance) assigned (created from) the data type (class) has the same characteristics as the data type

•Datatype
•Class

•Variable
•Instance

PL/SQL

```
v_first_name VARCHAR2(20) := 'Frank';
v_commission PLS_INTEGER := 200;
```

Java

```
String firstName = "Frank";
Integer salary = new Integer(200);
```

  - Difference: OO (Java) has methods for the declared instance

Code Demonstrations

- PL/SQL
  - Function code is **applied to** the variable

```
v_emp_name := UPPER(v_first_name);
v_initial := SUBSTR(v_first_name,
                    1, 1);
v_salary_char := TO_CHAR(v_salary);
```

- Java
  - Method code is **part of** the object

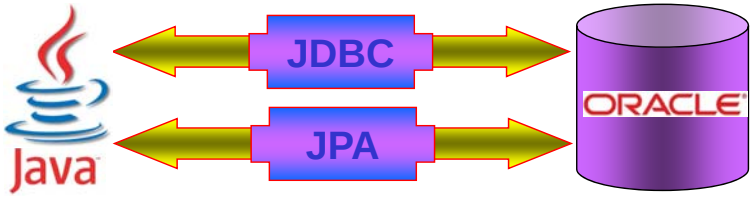
```
empName = firstName.toUpper();
initial = firstName.substr(0, 1);
salaryChar = salary.toString();
```


What Do You Really Need to Know?

- Java code organization and syntax
  - Piece of cake for C programmers
  - Smaller piece for PL/SQL enthusiasts
- Object orientation
  - Need to know inheritance well
    - WYSINAOWYG
      - What you see is not all of what you get
    - You can be effective in coding Java web apps with mastery of inheritance
  - Encapsulation and Polymorphism
    - Need to be able to recognize it and understand it

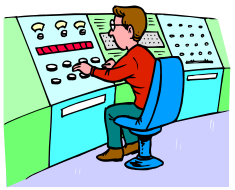
How Does Java Connect to the Database?

- It doesn't (on its own)
- Your Java program can call JDBC
  - Java Database Connectivity library
  - Can embed a SQL statement in the JDBC call
- Or use a JDBC framework
  - Enterprise JavaBeans (EJBs) – Java EE standard
  - ADF Business Components (Oracle ADF)
- Java Persistence API (JPA) framework to store data from Java objects



Agenda

- Foundations
- Parts list
- Language basics



Flon's Law

There is not now, and never will be, a language in which it is the least bit difficult to write bad programs.

Basic Java Terms

- Class
  - The "pattern" (building block)
  - Close parallel to PL/SQL package
- Object
  - An *instance* of a class
  - PL/SQL variable (on steroids)
- Method
  - Unit of code contained in a class
  - Like PL/SQL procedures and functions
- Constructor
  - Code unit used to instantiate an object
  - PL/SQL: used only for object types

About Methods

- Method signature:

Access specifier | Return type | Argument

```
public static void main (String[] args)
```

Does not require an object to use the method | Method name

- Access specifier declares which classes can see this method
  - E.g., "private" is not visible to other classes
- Return type can be something or nothing (void)
- Overloading allowed
  - More than one method with the same name and different arguments

Accessor Methods

- Used to affect private variables
  - Also called "getters and setters"
  - Like SELECT (getter) and UPDATE (setter)
  - Implement encapsulation
    - Protected access to private variables
- Example

```
...
private int height;
...
public void setHeight(int newHeight) throws Exception {
  if (newHeight < 1) {
    throw new Exception("Height must be a positive integer.");
  } else {
    height = newHeight;
  }
}
```

This protects the private variable value.

About Constructors

- Constructors look a bit like methods
- Name is the same as the class
- No return type (not even void)
  - Example signature:
    `Box(int quantity)`
- Responsible for instantiating the class
  - Creates the object
  - Use it to initialize variables
- Called using new operator:
  - **`Box usefulBox = new Box();`**
- There is a default (non-declared) constructor for every class

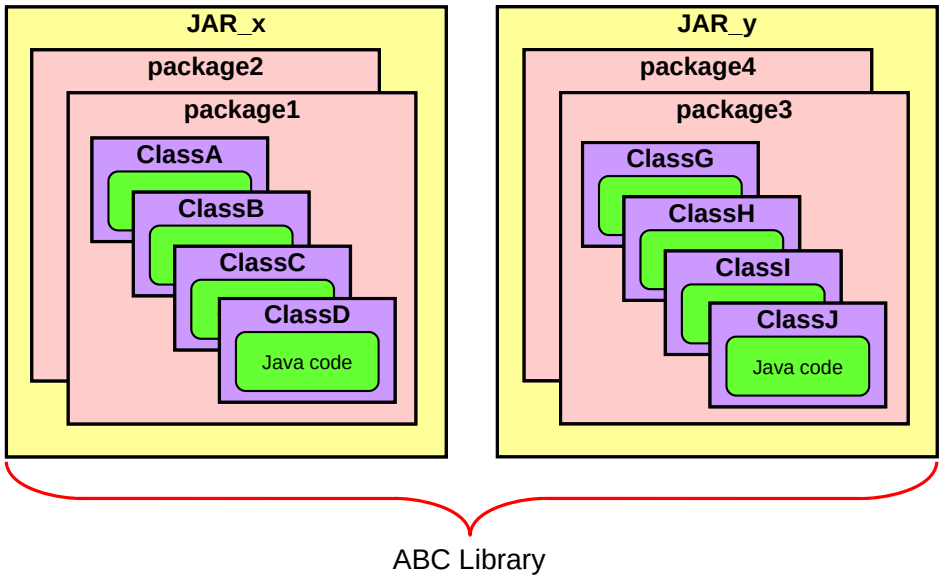
About Java Classes

- One “public” class per file
  - Public classes are available everywhere
- All code is contained in classes
- Each public class is stored in its own source file
  - Has exactly same name as class
  - File names are case sensitive
- Used to create objects, run code, or serve as superclasses

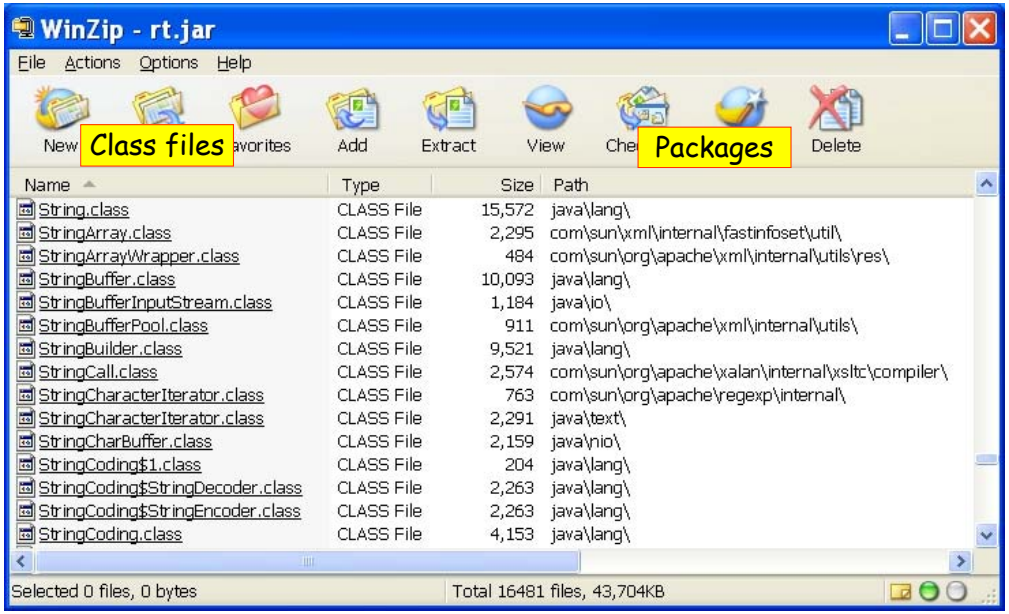
Classes, Packages, Libraries

- Class files contain Java code
- Packages are collections of class files
  - Directories in the file system
- Java Archive (JAR) contains multiple class files
  - Can use .jar or .zip extension
- Libraries are made of one or more JARs
  - Just a name applied to the collection of JAR files

Code Organization



Sample Archive Contents



About CLASSPATH

- JVM needs to be told where the classes are located
  - Your classes
  - Library classes
- Set the environment (shell) variable: CLASSPATH
  - The list includes directories or archive files (JAR or Zip), for example:

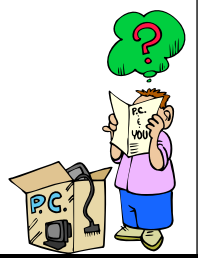
```
SET CLASSPATH=.;C:\JDev11\jdk\jre\lib\rt.jar
```

Current directory

Naming Conventions

- Java is a case-sensitive language
  - Keywords are in lower case
    - **`for, while, if, switch,`** etc.
    - Case compliance is enforced for keywords
- There are conventions for other names
  - Normally, no underscores used
    - For example, **`EmpLastName`** not **`EMP_LAST_NAME`**
  - Package names are all lower case
  - Class names are mixed case ("camel case")
    - **`EmployeeDataAccess`**
  - Method and variable names are init-lower
    - **`numberOfEmps, getCity()`**, **`setCity()`**
  - Constants use all uppercase and underscores
    - MAX_LOAD, MIN_HEIGHT

Note: Java names can have any number of characters.



Blocks and Comments

- Executable program blocks - { } symbols
  - Collection of declarations, specifiers, and methods
  - Code blocks can be nested
- Comments:
  - Single line

```
// This is a single-line comment.
int count;  // it can end a line
```

  - Multiline

```
/* This is a multiline comment in
  Java, the same as in SQL. */
/* It can be one line */
```

  - Javadoc

```
/** This will be written to an
    HTML document. */
```

Developing the Code

1. Create or modify source code file
   - Standard ASCII text – can use Notepad or vi
   - Use .java extension (HiThere.java)
2. Compile the source code file
   - **`javac.exe HiThere.java`**
   - Creates <classname>.class (HiThere.class)
3. Test the class file in a runtime (interpreter) session
   - Also called *Java Virtual Machine* (JVM)
   - **`java.exe HiThere`**
4. Repeat 1-3 until victory is declared
5. Deploy the file
   - Package with required libraries

Called "bytecode" or "bytecodes"

Example Class

Constructor

Code block symbol

accessor methods

Non-accessor method

```
package shapes;
public class Rectangle {
  private int height;
  private int width;
  public Rectangle() {
    height = 1;
    width = 1;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int newHeight) {
     height = newHeight;
  }
  public int getWidth() {
    return width;
  }
  public void setWidth(int newWidth) {
     width = newWidth;
  }
  public int area() {
     return height * width;
  }
}
```

Package statement

Class declaration

Variable declarations (attributes, fields)

Example Subclass

Subclass keyword

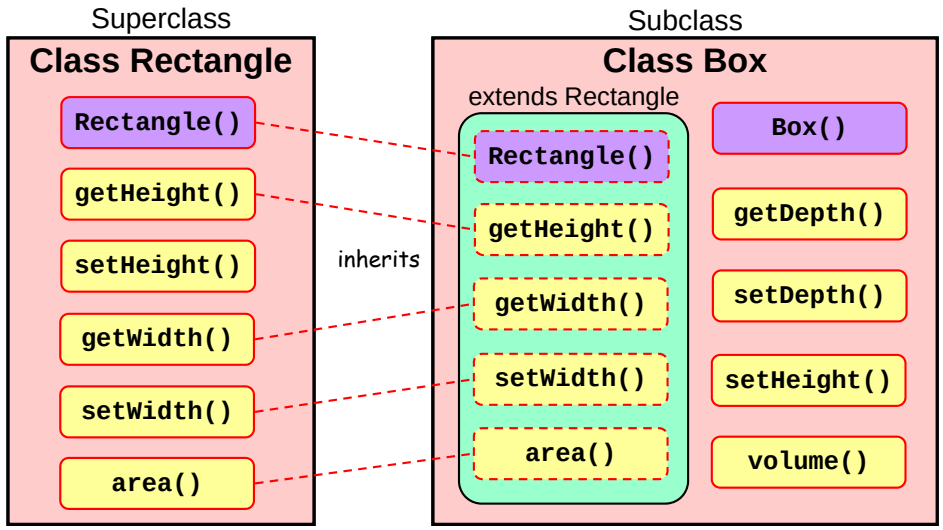
"this" is **this** (Box)
"super" is **that** (Rectangle)

Overrides super.setHeight()

Class import

```
package shapes;
import com.quovera.geometry.Polygon;
public class Box extends Rectangle {
  private Polygon shapePoly;
  private int depth;
  public Box() {
    setHeight(4);
    super.setWidth(3);
    this.depth = 2;
    shapePoly = new Polygon();
  }
  public int getDepth() {
    return depth;
  }
  public void setDepth(int newDepth) {
    depth = newDepth;
  }
  public void setHeight(int hght) {
    super.setHeight(hght * 2);
  }
  public int volume() {
    return area() * getDepth();
  }
}
```

Subclass Relationship



Using Box

```
public class TestBox {
  public static void main(String[] args) {
    Box usefulBox = new Box();
    System.out.println(
      "The height is " + usefulBox.getHeight() +
      ", depth is " + usefulBox.getDepth() +
      ", and width is " + usefulBox.getWidth());
    System.out.println(
      "The area is " + usefulBox.area() +
      " and volume is " + usefulBox.volume());
    usefulBox.setWidth(5);
    System.out.println(
      "Now area is " + usefulBox.area() +
      " and volume is " + usefulBox.volume());
  }
}
```

Object instantiation. Calls Box() which calls Rectangle()

Why is this method call special?

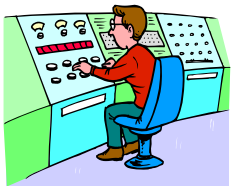
From Rectangle

From Box

```
C:\JavaSamples>java TestBox
The height is 8, depth is 2, and width is 3
The area is 24 and volume is 48
Now area is 40 and volume is 80
```


Agenda

- Foundations
- Parts list
- Language basics



Primitive Types - Number

- Whole number
  - **byte** (-128 to 127)
  - **short** (-32,768 to 32,767)
  - **int** (-2,147,483,648 to 2,147,483,647)
  - **long** (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
- Decimal place
  - **float** (3.4e-038 to 3.4e+038)
  - **double** (1.7e-308 to 1.7e+308)
    - More precise than float, but takes twice the memory space (64 bytes)

9.2 quintillion in North America
9.2 trillion in Europe and the UK

Primitive Types – Character and Logical

- Character
  - **char** (integer, 0 to 65,536)
  - Single character or symbol
  - Handles Unicode (an international character set)
- Logical
  - **boolean (true** or **false)**
  - Two values only (no null logical value)
  - **true** is not a number like –1
  - No quotes around the symbol
  - For example:

```
boolean isTrue = true;
isTrue = (2 < 1);
```

The only character primitive datatype.

Typing Based on a Class

- Use the new operator:

```
String testString;
testString = new String();
```

declaration

instantiation

OR

```
String somestring = new String();
```

Use StringBuffer() or StringBuilder() if you will change the data

- Most any class can be used to create an object
  - Exceptions: abstract classes, classes with private constructors
  - Data and behavior of the class are available to the object

Constants

- Use keyword **`final`**
  - Like **`CONSTANT`** in PL/SQL
  - Final variables must be initialized in same statement
  - For example,

```
static final double PI = 3.141592;
```

- Can use **`final`** for methods
  - Method cannot be overridden in a subclass
- Can use **`final`** for classes
  - Final **classes** cannot be inherited

Variable Scope

- Variables last within the curly brackets or structure that encloses them

```
for (int i = 0; i < 10; i++) {}
```

i available only during "for" loop

- Like PL/SQL nested blocks
- Curly brackets for if..else, loops count
  - Unlike PL/SQL

masterVar available here

ifVar not available here

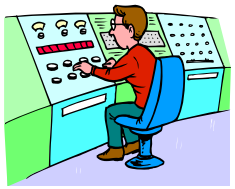
```
{
  int masterVar;
  if (true) {
    int ifVar;
  }
}
```

Some Java Operators

Function	PL/SQL	Java
Concatenation	||	+
Modulus (remainder)	MOD()	%
Assignment	:=	=
Increment	i := i + 1	i++
Addition assignment	i := i + 5	i += 5
Equal to	=	==
Not equal to	!=	!=
Logical AND	AND	&&
Logical OR	OR	||
Ternary if-then-else	DECODE()	? :
Bitwise unary not	[nothing]	~
Percolate	BREW()	☺

Standard Control Structures

- Sequence
  - Code executes in the order in which it is written
- Conditional branching
  - **`if else, switch`**
- Iteration
  - **`while, for, do while`**
- Jump statements
  - **`break`** – to exit a structure like switch
  - **`continue`** – to start loop over
  - **`return`** – to go back to calling routine
  - No **`goto`**
- Exception handling
  - Enclose in **`try {} catch {}`** block
  - **`throw`** causes an explicit exception
  - Like PL/SQL **`BEGIN..EXCEPTION..END`**



Conditional Example

```
class ShowQuarter {
  public static void main (String[] args) {
    int taxMonth = 10;
    String taxQuarter;

    if (taxMonth == 1 || taxMonth == 2 || taxMonth == 3) {
      taxQuarter = "1st Quarter";
      }
    else if (taxMonth >= 4 && taxMonth <= 6) {
      taxQuarter = "2nd Quarter";
      }
    else if (taxMonth >= 7 && taxMonth <= 9) {
      taxQuarter = "3rd Quarter";
      }
    else if (taxMonth >= 10 && taxMonth <= 12){
      taxQuarter = "4th Quarter";
      }
    else {
      taxQuarter = "Not Valid";
      }
    System.out.println("Your current Tax Quarter is: " +
      taxQuarter );
  }
}
```

comparison equals

Logical OR

Logical AND

Loop Examples

```
class DemoFor {
  public static void main (String[] args) {
    int i;
    for (i = 1; i <= 10; i++) {
      System.out.println("Loop count is " + i);
    }
  }
}
```

println() handles mixing of data types

increment operator

Alternative: `for (int i = 1; i <= 10; i++)`

```
class DemoWhile {
  public static void main (String[] args) {
    int i = 1;

    while (i <= 10) {
      System.out.println(
        "Loop count is " + i);
      i++;
    }
  }
}
```

```
Loop count is 1
Loop count is 2
Loop count is 3
Loop count is 4
Loop count is 5
Loop count is 6
Loop count is 7
Loop count is 8
Loop count is 9
Loop count is 10
```

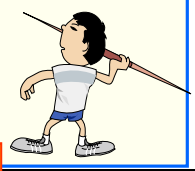

Exception Handling Example

```
public class TestException extends Object {
  public static void main(String[] args) {
    int numerator = 5, denominator = 0, ratio;
    try {
      ratio = numerator / denominator;
      System.out.println("The ratio is " + ratio);
    }
    catch (SQLException sqlexcept) {
      // display SQL error message
      sqlexcept.printStackTrace();
    }
    catch (Exception except) {
      // display generic error message
      except.printStackTrace();
    }
    finally {
      System.out.println("After finally.");
    }
    System.out.println("The end.");
  }
}
```

Will throw a divide-by-zero error.

Always run

Not run if an unhandled exception occurs



Some Gotchas

- Automatic rounding – use casting or correct datatype
  - E.g., the result of an int/int is an int:

```
int numA = 2, numB=3;
System.out.println("result=" + numB/numA);
```

result=1

- When in doubt of order of precedence, use ()

```
System.out.println("result1=" + 4 + 5);
System.out.println("result2=" + (4 + 5));
```

result1=45
result2=9

- Use **{}** around all if statement clauses
  - Only one statement executes after **if**

```
if (anInteger == 0)
  anotherInteger = 5;
  someInteger = 10;
```

This always executes

- Use equals to compare objects
  - Instead of:

```
if ( newBox.width == 4 )
```

Could throw an NPE

  - Use:

```
if ( newBox.width.equals(4) )
```

Bibliography

- *Java, The Complete Reference, 9th Ed*
  - Herb Schildt, Osborne McGraw-Hill
- *Thinking in Java* (online or hard copy)
  - Bruce Eckels, www.mindview.net
- *Head First Java*
  - Kathy Sierra, Bert Bates, O'Reilly
- *Java Tutorials*
  - docs.oracle.com/javase/tutorial/
- *Refactoring: Improving the Design of Existing Code*
  - Martin Fowler, Addison-Wesley

Quote

Real programmers
can write assembly code
in any language.

—Larry Wall

Summary

- Java has the basic language elements
- Java is a case-sensitive language
- A class is the building block
- An object is created from a class; like a PL/SQL variable typed from a datatype
- Objects are typed from primitive data types or from classes
- Recognized naming conventions
- Other than syntax, the big difference between Java and PL/SQL is OO

Quiz at the end

ODTUG Kscope17

SAN ANTONIO, TEXAS * JUNE 25-29

Please fill out the evals

Some of the 8 books co-authored with Dr. Paul Dorsey, Avrom Roy-Faderman, & Duncan Mills

quovera

http://www.quovera.com

- **Founded in 1995 as Millennia Vision Corp.**
- **More technical white papers and presentations on the web site**

United Nations Translation Exercise

You are translating "Scott's" speech for "Larry." * What do you say when Scott says:

- { }
- class
- object
- inheritance
- static
- package
- final
- constructor
- method
- public
- private
- Box usefulBox = new Box();
- usefulBox.getMHeight();
- throw

* Any resemblance to Larry Ellison (Oracle) and Scott McNealy(formerly of Sun Microsystems formerly charge of Java), is purely coincidental.

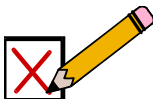
Quiz Time

1. Is **`for`** and **`FOR`** the same in Java?
2. How do you define a procedure in Java?
3. What is a constructor?
4. What's wrong with the following Java?

```
string EmpName = 'SCOTT';
```

5. What does this mean?

```
public static void main()
```



Quiz Time

6. What does the symbol “Test” represent in the following?

```
public class Test
public Test()
public void test()
int test;
```

7. What is the main container for Java code?
8. What is a subclass?
9. What is a method?

Quiz Answers

Scott Says	You Tell Larry	Scott Says	You Tell Larry
{}	BEGIN END	constructor	Initialization section in a PL/SQL package body
class	PL/SQL package	method	Function or procedure
object	variable	public	Grant to PUBLIC
inheritance	(nothing unless Larry knows object extensions in Oracle)	private	Not granted, or package body only
static	global	Box usefulBox = new Box();	v_useful_box NUMBER;
package	directory	usefulBox.getMHeight();	my_package. my_function();
final	CONSTANT	throw	RAISE

Quiz Answers

1. No. Java is case sensitive.
2. Java only has methods not procedures, but a method with a void return corresponds to a PL/SQL procedure.
3. It is a code unit used to create an object.
4. You need to use double quotes to delimit strings. Also, by convention, variables are initial lowercase. In addition, the standard string class is spelled “String”.
5. The method, main, is available to all callers (public). It returns nothing (void) and you do not need an object to call it (static). In addition, if this signature had a String array for an argument, you could call main() from the command line by running the class.

Quiz Answers

6. These are, a class, a constructor, a method, and a variable, respectively.
7. The class is the direct container for Java code.
8. A subclass is a class declared as a child of another class using the keyword “extends.”
9. A method is the main code unit in Java. Methods are contained in class files.