


## Java Language Essentials

- ❖ Java is **Case Sensitive**
  - ◆ All **Keywords** are lower case
- ❖ White space characters are ignored
  - ◆ Spaces, tabs, new lines
- ❖ Java statements usually end with a semicolon ;
- ❖ Java compound statements use { }
- ❖ Java Language Comments
  - ◆ Single-line // Comment goes to end of line
  - ◆ Multi-line comments /\* This is a comment \*/
  - ◆ Create a title block at beginning of program to describe program
  - ◆ Describe purpose of unusual code
- ❖ Use descriptive identifiers



Copyright © 2006 R.M. Laurie 1

## Java Identifiers

- ❖ Identifier Naming Rules
  - ◆ Identifiers are case sensitive
  - ◆ Begin each identifier with a letter, \_ underscore, or \$ dollar sign (no numbers)
  - ◆ May use numbers for any character after first
  - ◆ No spaces allowed, but may use underscore
  - ◆ Limit Identifier length to 20 characters (Max=255)
  - ◆ Not a Java keyword (approximately 50)
  - ◆ **public class name** must be same as *filename.java*
- ❖ Identifier Naming Guidelines
  - ◆ **Class identifier** should be TitleCase without spaces
  - ◆ **Method identifier** should start with lower case and be a verbNoun combination with words in title case
  - ◆ **Variable identifiers** are nouns and begin with lower case letter(s) to indicate variable data type

Copyright © 2006 R.M. Laurie 2

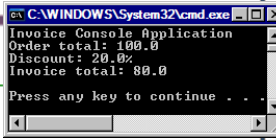
## Java Console Application

```

/* Invoice Console Application
 * Author: Robert Laurie
 * Date: 1 February 2006
 *
 * -----
 * Description: This program will
 * calculate the invoice price based
 * on a 20% discount
 */
public class InvoiceConsoleAp
{
    public static void main(String[] args)
    {
        // VARIABLE DECLARATIONS
        double dDiscount, dInvoice, dOrder = 100;

        // PROCESSING
        dDiscount = dOrder * 0.2;
        dInvoice = dOrder - dDiscount;

        // DISPLAY RESULTS
        System.out.println("Invoice Console Application");
        System.out.println("Order total: " + dOrder);
        System.out.println("Discount: " + dDiscount + "%");
        System.out.println("Invoice total: " + dInvoice + "\n");
    }
}
    
```



Copyright © 2006 R.M. Laurie 3

## Java Keywords

boolean	private	for	transient
char	protected	continue	instanceof
byte	public	do	true
float	static	extends	false
void	new	class	throws
short	this	volatile	native
double	super	while	implements
int	interface	return	import
long	package	throw	synchronized
abstract	switch	try	const
if	case	catch	goto
else	break	finally	null
final	default		

Copyright © 2006 R.M. Laurie 4

## *Java Applications have Class*

- ❖ **Java applications are contained in classes**
  - ◆ Each *filename.java* file must have one (and only one) **public class** declaration with same name:
 

```
public class filename // Class Header Line
{ // Class definition body begins here
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java");
    } // Class definition body ends here
}
```
- ❖ **Every Java applications contain methods**
  - ◆ **main()** method is start of program for java
  - ◆ **public** keyword means other classes can access main method
  - ◆ **static** keyword means method can be called from other classes
  - ◆ **void** keyword means method won't return any values
  - ◆ Every main has an argument **args** defined as array of strings

Copyright © 2006 R.M. Laurie | 5

```
public class ShowMessageCls
{
    // Class data declaration section
    private String sMessage;
    //Class method definition section
    ShowMessageCls()
    {
        sMessage = "I need a cup of Java.";
    }
    public void displayMessage()
    {
        System.out.println(sMessage);
    }
}
```

```
public class ShowMessagePrg
{
    public static void main(String[] args)
    {
        // Create a variable of type ShowMessageCls
        ShowMessageCls oMessageOne;
        // Create an object of the ShowMessageCls
        oMessageOne = new ShowMessageCls();
        // Call the method for the object
        oMessageOne.displayMessage();
    }
}
```

Source program

↓

Compiler

↓

Bytecode files

↓

Libraries

↓

Java Virtual Machine (JVM) Interpreter

↓

Executing program

Copyright © 2006 R.M. Laurie | 6

## *PrintStream Class Methods*

- ❖ **PrintStream class, methods:**
  - ◆ **print()** // Std output and ends with new line
  - ◆ **println()** // Std output and no new line
  - ◆ **\n** is a new line character in string "Hello \nWorld"
- ❖ **Package**
  - ◆ Multiple classes stored in same directory
    - ◆ **ShowFirstMessage** and **UseShowFirstMessage**
- ❖ **General syntax:**
  - ◆ `objectName.print(data)`
  - ◆ `System.out.print("Hello World!");`
- ❖ **Method Input Data**
  - ◆ **Parameters** = What input does a method need?
  - ◆ **Argument** = Actual data Item passed to method

Copyright © 2006 R.M. Laurie | 7

## *System Class*

- ❖ **Provides methods for examining system-related information such as:**
  - ◆ Name of operating system
  - ◆ Java version number
- ❖ **Supports basic input and output**
  - ◆ `System.out.print( "Hello " );`
  - ◆ `System.out.println( "World!" );`
- ❖ **Sources for documentation:**
  - ◆ <http://java.sun.com/docs/search.html>
  - ◆ Same as the file that we downloaded and unzipped in class

Copyright © 2006 R.M. Laurie | 8