Java For The Beginner
Part II of III

Theresa Tai
IBM System z New Technology Center
Poughkeepsie, New York
ttai@us.ibm.com

February 25, 2008
Session 8353

Housekeeping Reminder

- No food or drink in the Lab
- Please silent mobile phones
- Don't hesitate to ask questions
- Have fun!

Agenda

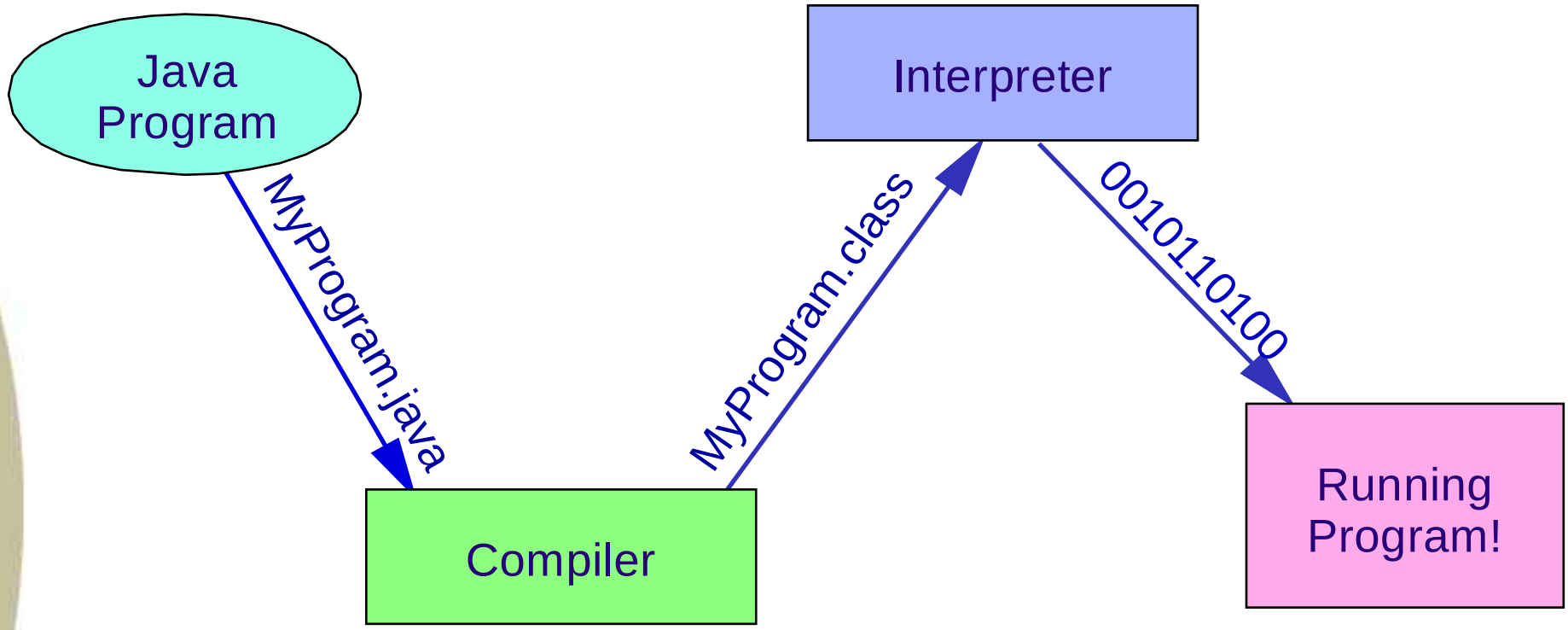
- Lecture
  - What is Java?
  - Java Basics
  - Java Code Structure
- Hands-on Lab
  - Explore the Eclipse Development Environment
  - Write and Run Simple Java programs

What is Java?

- A platform
  - Software only
  - Runs on top of hardware platforms
  - Two components:
    - JVM – Java Virtual Machine
    - API – Application Programming Interface
- A programming language
  - Compiled and Interpreted
- Java software platform consists of
  - The Java language, JVM and Java class libraries

What is Java Language?

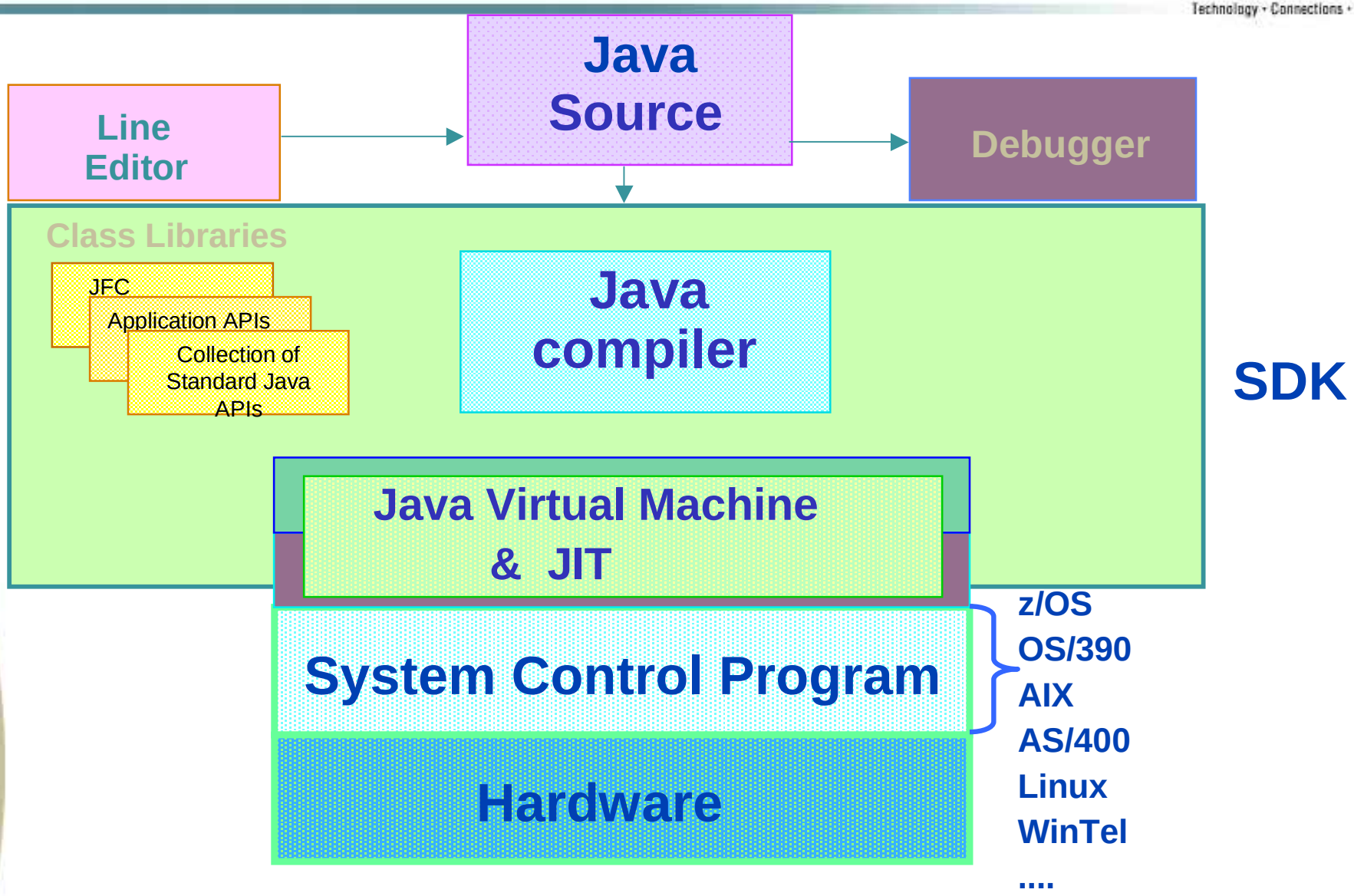
- A programming language (has some of the characteristic as C++)
  - Source code in plain text files with a .java extension
- Compiled and interpreted
  - .java source files compiled into .class files



Java Bytecodes

- Instructions for the Java Virtual Machine
- Write Once, Run Anywhere
  - Compiled bytecode is platform independent
  - Any device capable of running Java will be able to interpret bytecode into platform specifics
- Development Tools
  - The Java compiler (javac)
  - The Java launcher (java)
  - The Java documentation tool (javadoc)

The Java Platform



Benefits of Java

- Get started quickly
- Write less code
- Write better code
- Write programs faster
- Avoid platform dependencies
- Write once, run anywhere
- Distribute software more easily
- Network enabled

The Java APIs and Integration Libraries

- Application Programming Interfaces (APIs)
  - Provides the core functionality of the Java programming language
  - A set of class libraries
    - From basic objects, to networking and security, XML generation and database access
    - Programmers uses when writing Java source code
- Included in Java platform
- Prewritten code
  - Organized into packages of similar topics
- Integration Libraries
  - IDL, JDBC, JNDI, RMI and RMI-IIOP
  - Enable database access and manipulation of remote objects

The Core API – The Essentials

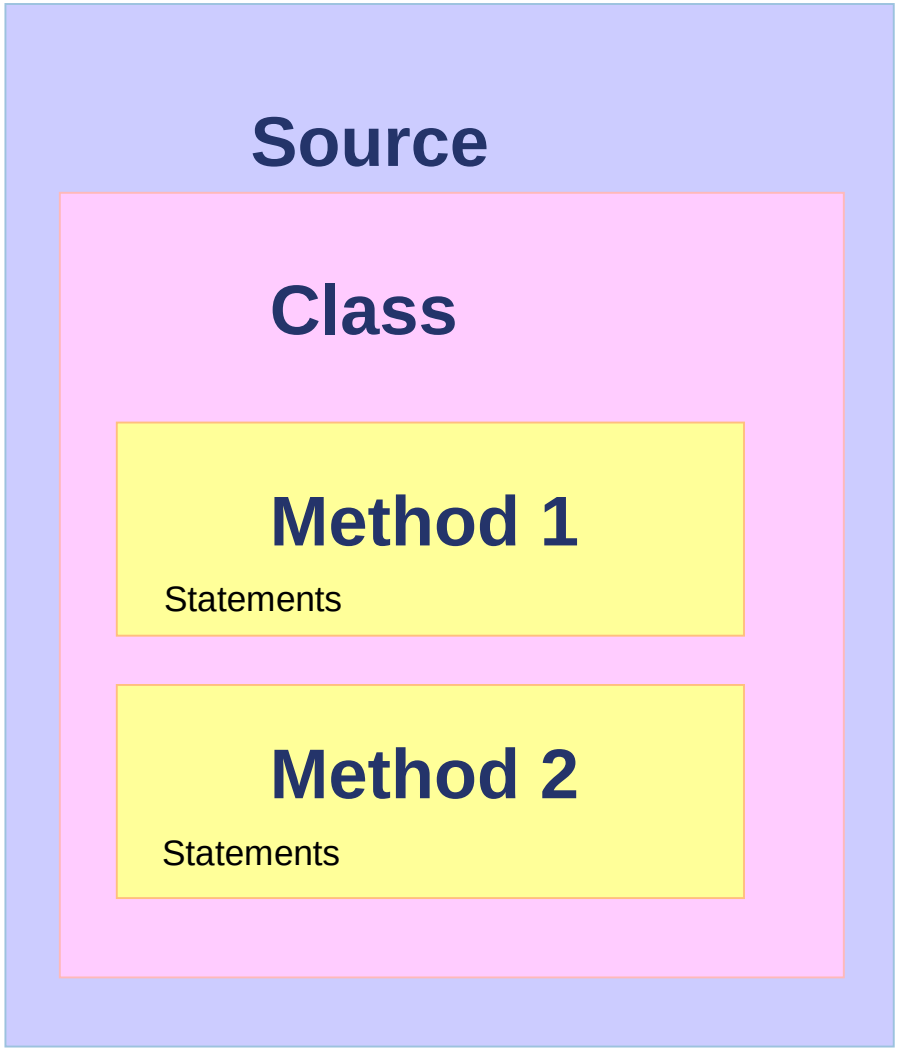
- Objects
- Threads
- Input and output
- System properties
- Strings
- Numbers
- Data structures
- Date and time

More API Packages

- Applets
- Internationalization
- Security
- Graphical User Interface
- Serialization
- Java Database Connectivity (JDBC)

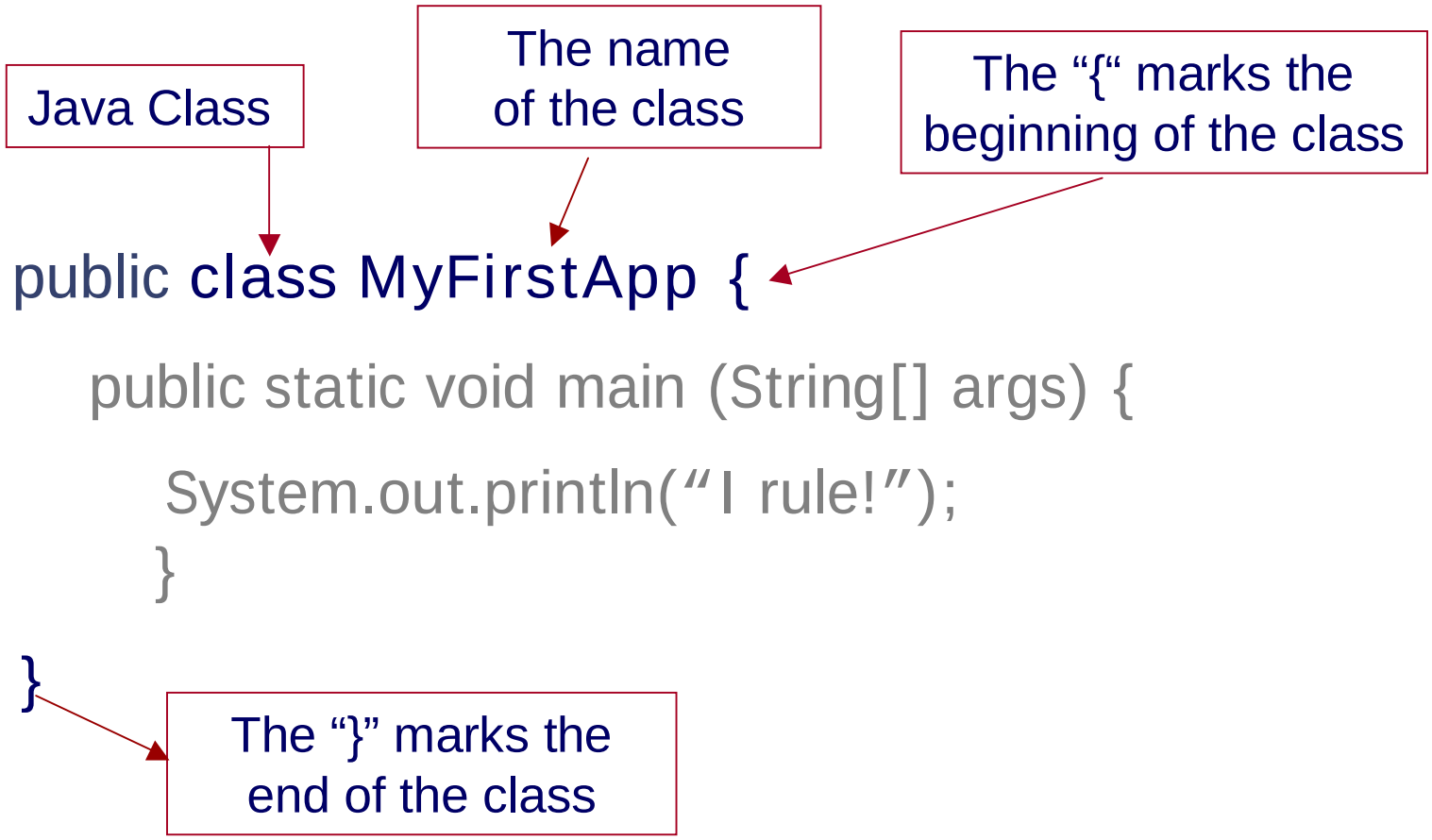
Java Code Structure

- Source file
  - Java source code
  - .java file extension
  - Holds class definition
- Class
  - A construct defines data and methods
  - One or more methods
- Methods
  - One or more sequence of statements
- Statements
  - Typically operate on data



Anatomy of a class

The fundamental building block in Java programming language is the ***class***

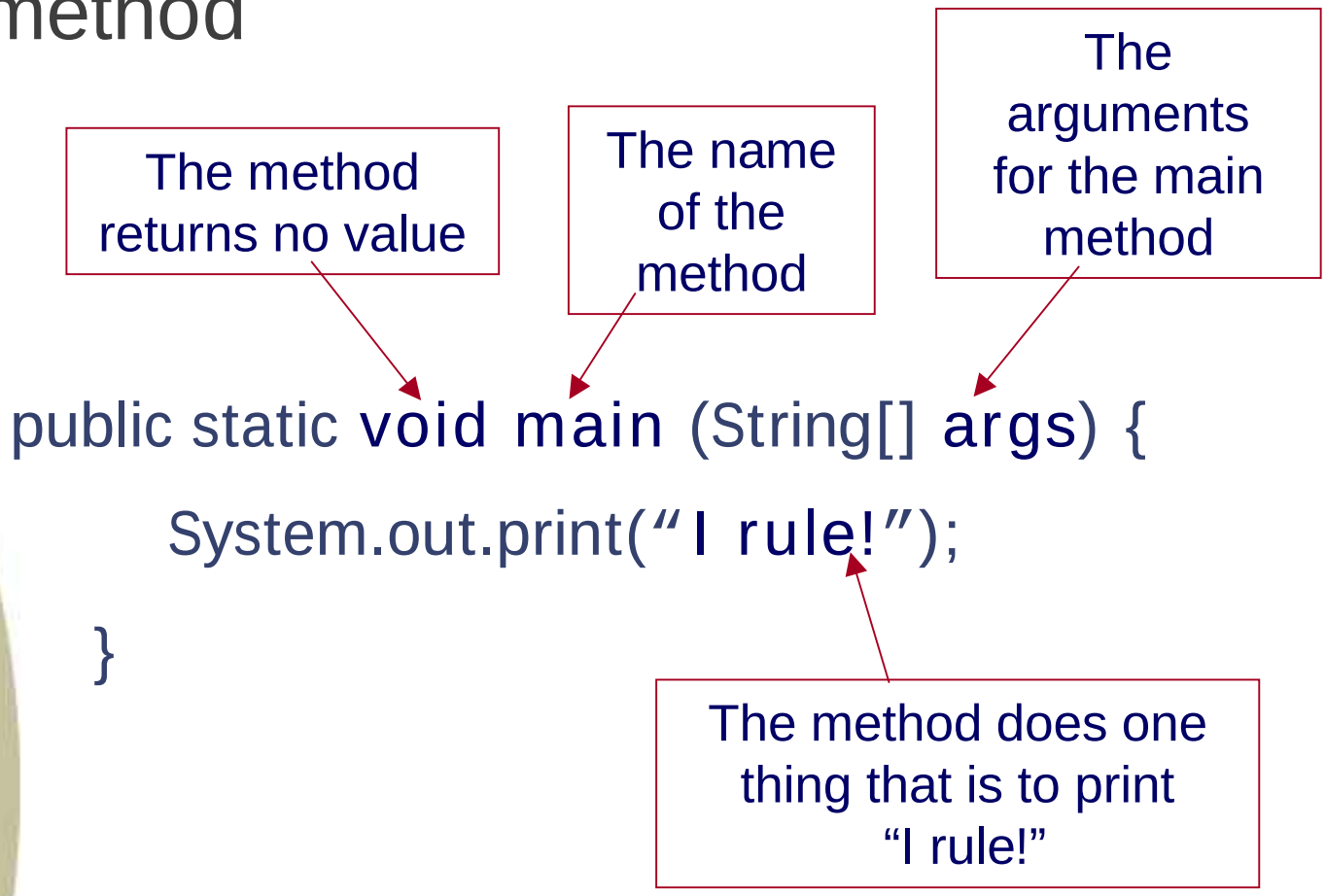


Java Class

a class in and of itself is not an object… Its like a blueprint that define how object will look and behave when the object is created or instantiated from the specification declared by the class… just as you can construct many houses all the same from the same blueprint/architecture drawing.

Anatomy of a main Method

The entry point to every application is its ***main*** method



Basic Java Syntax

- Comments
- Variables and Data Types
- Primitive Data Types
- Reference Data Types
- Operators
- Expressions
- Arrays
- Strings

Comments

- /* text */
  - Java supports the familiar C-style comments /* text */
  - The Java compiler ignores everything from /* to */
- /** documentation */
  - A documentation or “doc” comment, used by the javadoc tool
- // text
  - The compiler ignores everything to the end of the line

Variables and Data Types

- Variable declaration
  - Name
    - Can begin with letter, dollar sign, or underscore
    - Followed by letters, underscores, dollar signs, or digits
    - Convention is Upper case
  - Type
    - Java's compiler cares about type
    - Determines value and operations
- Two kinds of variables
  - Primitive
  - Object Reference

Primitive Types

- Hold fundamental values (simple bit patterns)
  - Numeric data types
    - Integers
      - *8-bit byte*
      - *16-bit short*
      - *32-bit int*
      - *64-bit long*
    - Floating point numbers
      - *Real numeric types are 32-bit float and 64-bit double*
  - Booleans
    - “TRUE”, “FALSE”, “YES”, “NO” or similar constructs
  - Characters
    - Char myChar = ‘A’;

Primitive Types

Type	Bit Depth	Value Range
boolean	varies	true or false
char	16 bits	0 to 65535
byte	8 bits	-128 to 127
short	16 bits	-32768 to 32768
int	32 bits	-2147483648 to 2147483647
long	64 bits	-huge to huge
float	32 bits	varies
double	64 bites	varies

Reference Types

- Anything that is not primitive
  - Objects such as
    - Strings
    - Arrays
    - Classes
    - Interfaces

Operators - Arithmetic

Operator	Use	Description
+	op1 + op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes remainder of dividing op1 by op2

Operators – Increment / Decrement

Operator	Use	Description
++	op++	Increments op by 1; evaluates to the value of op before it was incremented
++	++op	Increments op by 1; evaluates to the value of op after it was incremented
--	op--	Decrements op by 1; evaluates to the value of op before it was decremented
--	--op	Decrements op by 1; evaluates to the value of op after it was decremented

Operators – Relational

Operator	Use	Returns **true** if
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Operators – Conditional

Operator	Use	Returns **true** if
&&	op1 && op2	op1 and op2 are both true, conditionally evaluates op2
||	op1 || op2	either op1 or op2 is true, conditionally evaluates op2
!	! Op	op is false
&	op1 & op2	op1 and op2 are both true, always evaluates op1 and op2
|	op1 | op2	either op1 or op2 is true, always evaluates op1 and op2
^	op1 ^ op2	if op1 & op2 are different – that if one or the other of the operands is true but not both

Operators - Assignment

Operator	Use	Equivalent to
=	op1 = op2	Assign op1 to the value in op2
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2

Expressions

- Series of variables, operations and method calls that evaluate to a single expression
  - Use parenthesis to specify *precedence*

```
int someNum = 6;
int anotherNum;
anotherNum = someNum –1;
anotherNum = (someNum – 3) * 2;
anotherNum = someNum – 3 * 2;
```

Strings

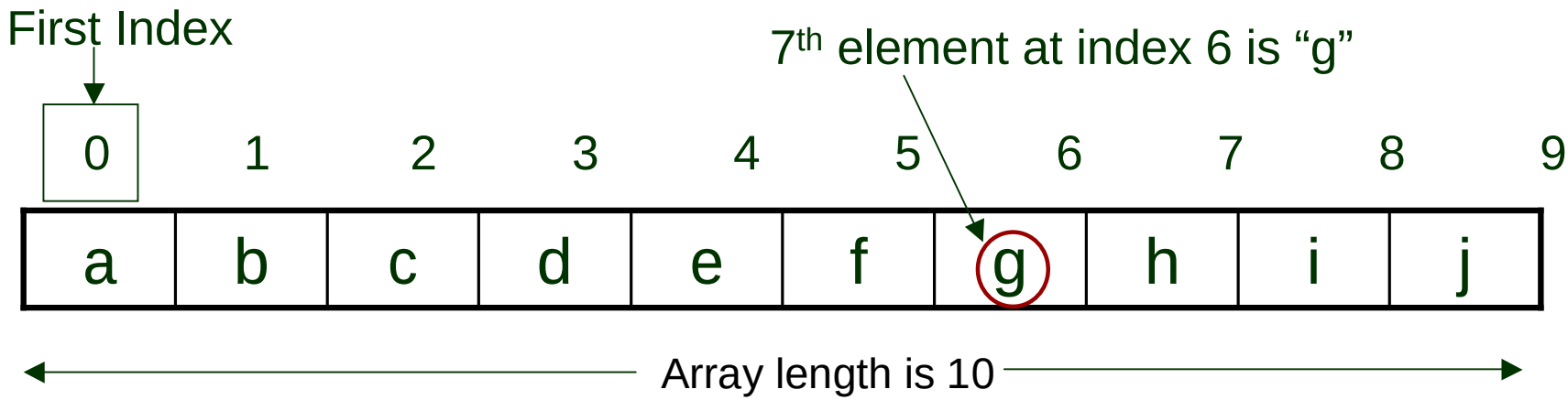
- The String class is included in the
  ➔ java.lang.Object package
- The String class represents character strings
- When you declare and use a String, you are actually using an instance of the String class
- Basic use of a String

```
String s = “Hello World! ”;
String t = “Look at Me.”;
System.out.println(s + t);
```

Hello World! Look at Me.

Arrays

- Array class is included in the java.lang.Object package
- The Array class contains various methods for manipulating arrays
- Access array elements using [ ] → anArray[0] = 10;
- Special array property length → anArray.length
- Declare as *type[ ]* varName; → int[ ] myInts;
- Must allocate memory before use → myInts = new int[10];
- General form → elementType[ ] arrayName=new elementType[arraySize];



Array Sample Code


```
int [ ] squares = new int[5];  // create an array of integers

squares[0] = 100;              // initialize first element
squares[1] = 200;              // initialize second element
squares[2] = 300;
squares[3] = 400;
squares[4] = 500;
```

If ... then ...

```
if (boolean_expr)
{then_stmnts;
}
```

```
if (a > 10) {
    System.out.println("a > 10");
}
```

- Test against boolean expression (not integer as in C/C++).
- Curly braces delimit blocks of code {}
- If the condition is true then the statements in the then block are executed.

If ... then ... else

```
if (expr) {
    then_stmnts;
}
else {
    else_stmnts;
}
```

```
if (a < 10) {
    System.out.println("a < 10");
}
else {
    System.out.println("a >= 10");
}
```

if the condition is false, then the statements in the else block are executed.

Nested If ... then ... else

```
if (expr) {
    then_stmnts;
}
else if (expr_1) {
    else_stmnts;
}
else {
    else_stmnts;
}
```

```
int testscore;  (int testscore=88;)
char grade;

if (testscore >= 90) {
    grade = 'A';
} else if (testscore >= 80) {
    grade = 'B';
} else if (testscore >= 70) {
    grade = 'C';
} else if (testscore >= 60) {
    grade = 'D';
} else {
    grade = 'F';
}
```

The Switch Statement

- Used to make a choice between multiple alternative execution paths
- A switch works with the byte, short, char or int primitive data types
  - A switch also works with enumerated types

Switch

```
switch ( int_expr ) {
    case x:
        case x stmnts;
        break;
    case y:
        case y stmnts;
        break;
    case z:
        case z stmnts;
        break;
    default:
        default case stmnts;
        break;
}
```

```
switch ( grade ) {
    case 'A':
        System.out.println("Outstanding!");
        break;
    case 'B':
        System.out.println("Well done");
        break;
    case 'C':
        System.out.println("Satisfactory");
        break;
    default:
        System.out.println("Fail");
        break;
}
```

Switch with “break”

- The “default” case is optional
- The “break” statement is optional, if it is omitted, execution drops through to the next case
  - A common source of errors!

```
switch ( grade ) {
        case 'A':
        case 'B':
        case 'C':
                System.out.println("Pass");
                break;
        default:
                System.out.println("Fail");
                break;
}
```

About Eclipse Tooling

- Free download from http://www.eclipse.org
  - Click on Download Eclipse
  - The Java for the Beginner Labs use the Eclipse IDE for Java Developers

  - IBM development tooling such as WSAD, WSDD, WSAD/IE and RAD are extensions to Eclipse
- Eclipse is an excellent starting point for learning Java development on your own
  - Start with workbench basics and tutorials
- Eclipse tool hints and tips
  - Click Help > “Tips and Tricks” from the menu bar

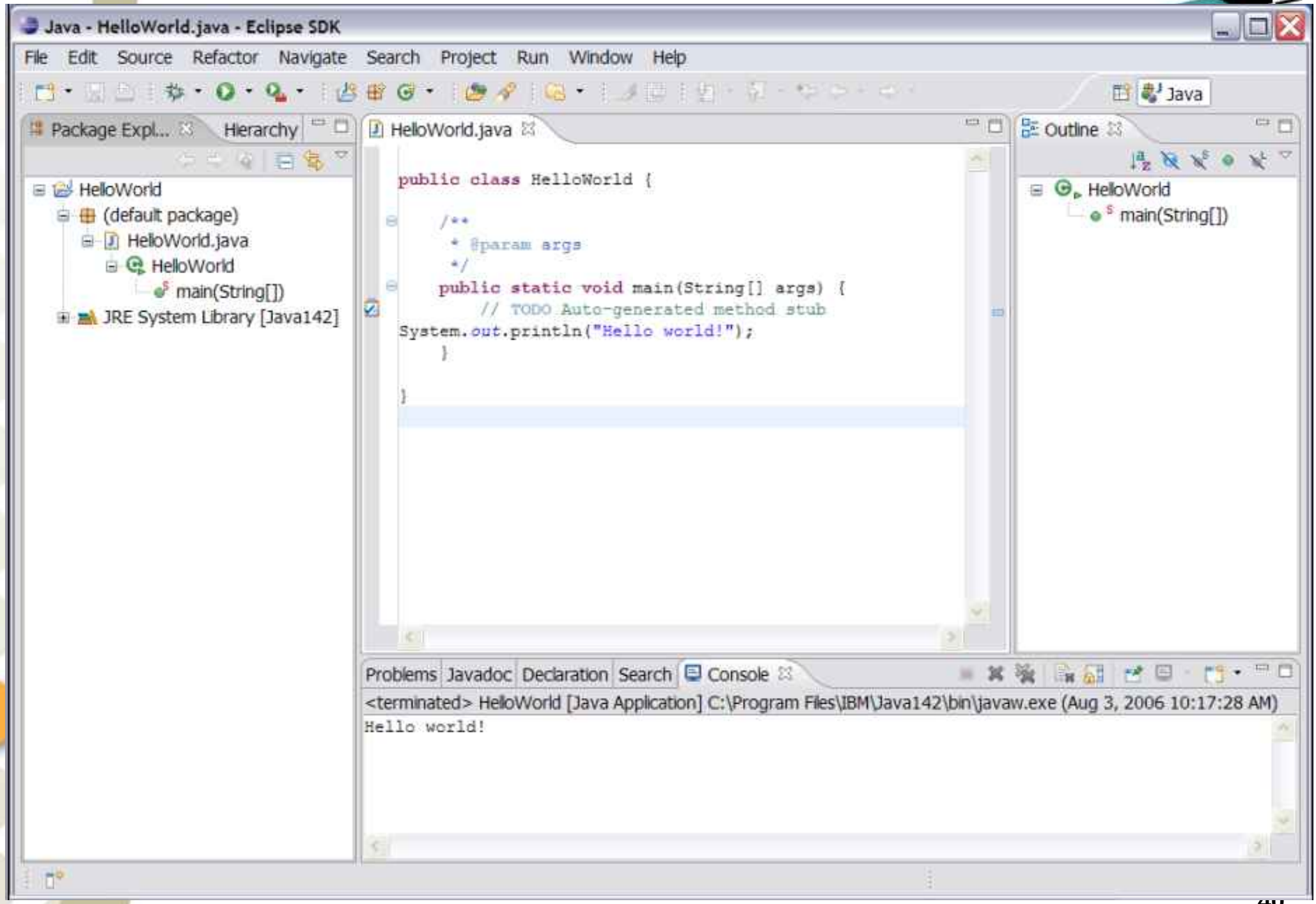
Lab Exercise

- Hello World
- Marathon

Please follow the Lab instructions, Have fun!
and Feel Free to Ask Questions…

Ex. 1 Hello World (sample solution)

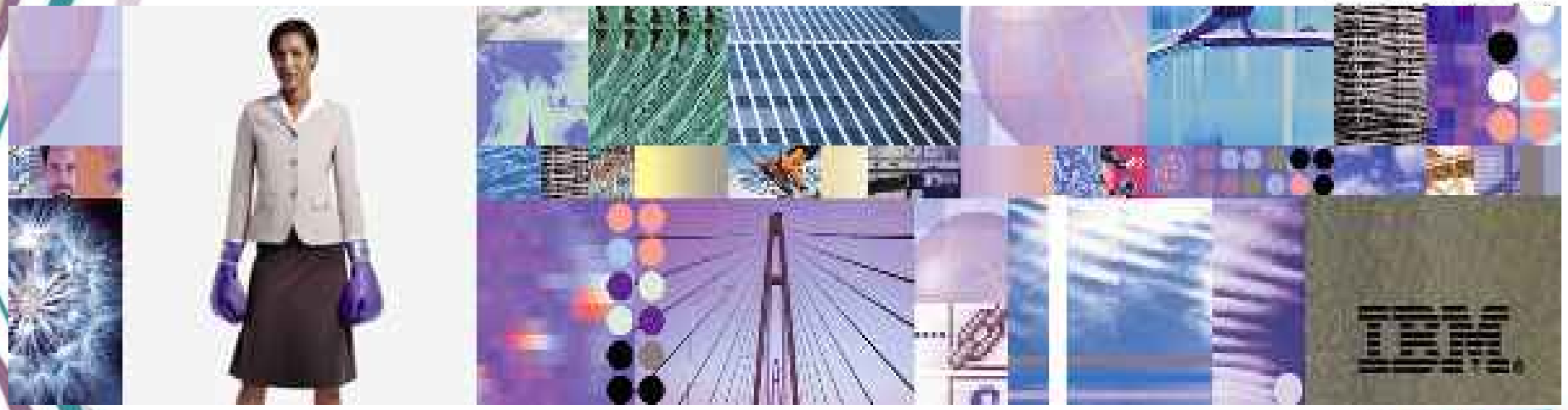
```
/**
 * The HelloWorldApp class implements an
   application that
 * simply displays "Hello World!" to the standard
   output.
 */
class HelloWorldApp {
public static void main(String[] args) {
   System.out.println("Hello World!"); //Display the
   string.
  }
}
```


Java - HelloWorld.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Expl... Hierarchy
HelloWorld
(default package)
HelloWorld.java
HelloWorld
main(String[])
JRE System Library [Java142]
HelloWorld.java
public class HelloWorld {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Hello world!");
}
}
Outline
HelloWorld
main(String[])
Problems Javadoc Declaration Search Console
<terminated> HelloWorld [Java Application] C:\Program Files\IBM\Java142\bin\javaw.exe (Aug 3, 2006 10:17:28 AM)
Hello world!

Ex. 2 Marathon (sample solution)


```
class Marathon {

public static void main(String[] args) {
   // TO DO:
   // Declare two integer variables, miles and yards, and one double variable,
   int miles;
   int yards;
   double ;
    // TO DO:
   // Set miles to 26, and yards to 385
   miles = 26;
   yards = 385;
   // TO DO:
   // Write an expression to calculate  from miles and yards.
   // Save the result of the expression in the variable .
   // One mile is 1.609 and there are 1760.0 yards in a mile
    = 1.609 * (miles + (yards / 1760.0));
   // Print the answer
   System.out.println("A marathon is " +  + " .");
  }  // end of main method
 } // end of marathon class
```



Thank You