Java and Eclipse for Beginner Part I
Session 1181

Presented by

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





Housekeeping Reminder

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

Content

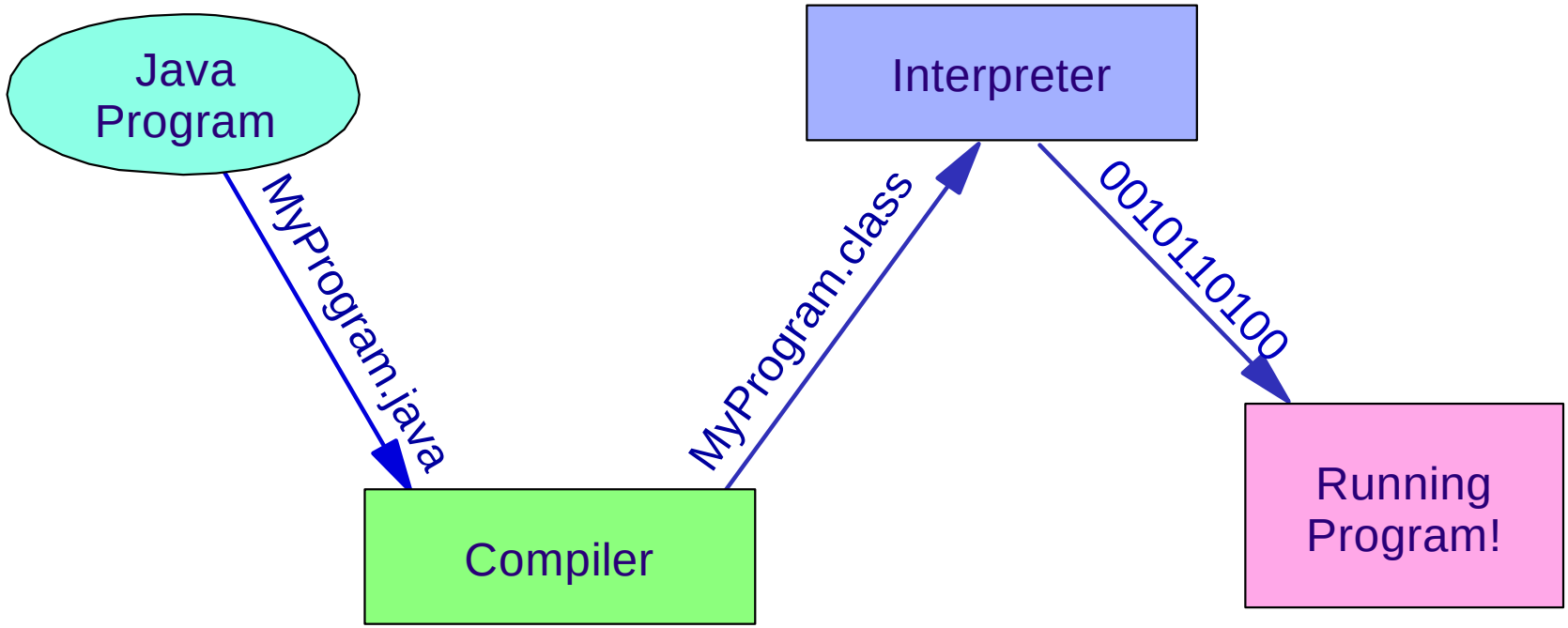
- Lecture
  - What is Java?
  - Java Basics
  - Java Code Structure
  - Eclipse Basics
- Hands-on Lab
  - Explore the Eclipse Development Environment
  - Write and Run Simple Java Applications

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?

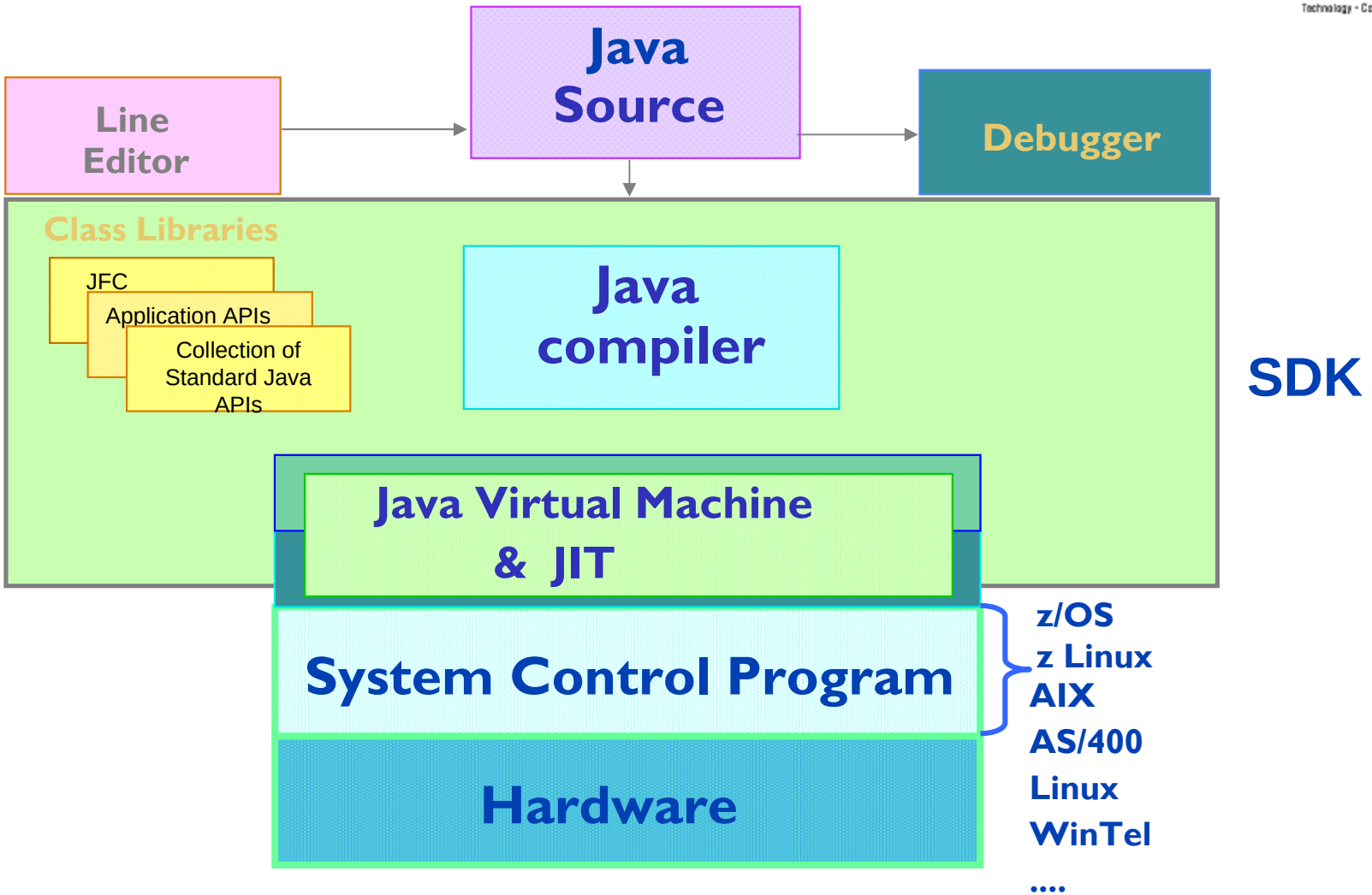
- A programming language (has some of the characteristic as C++)
  - Source code in plain text files with a .java extension
- Compiled and interpreted
  - .java source files compiled into .class files



Java Bytecodes

- Instructions for the Java Virtual Machine (JVM)
- 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

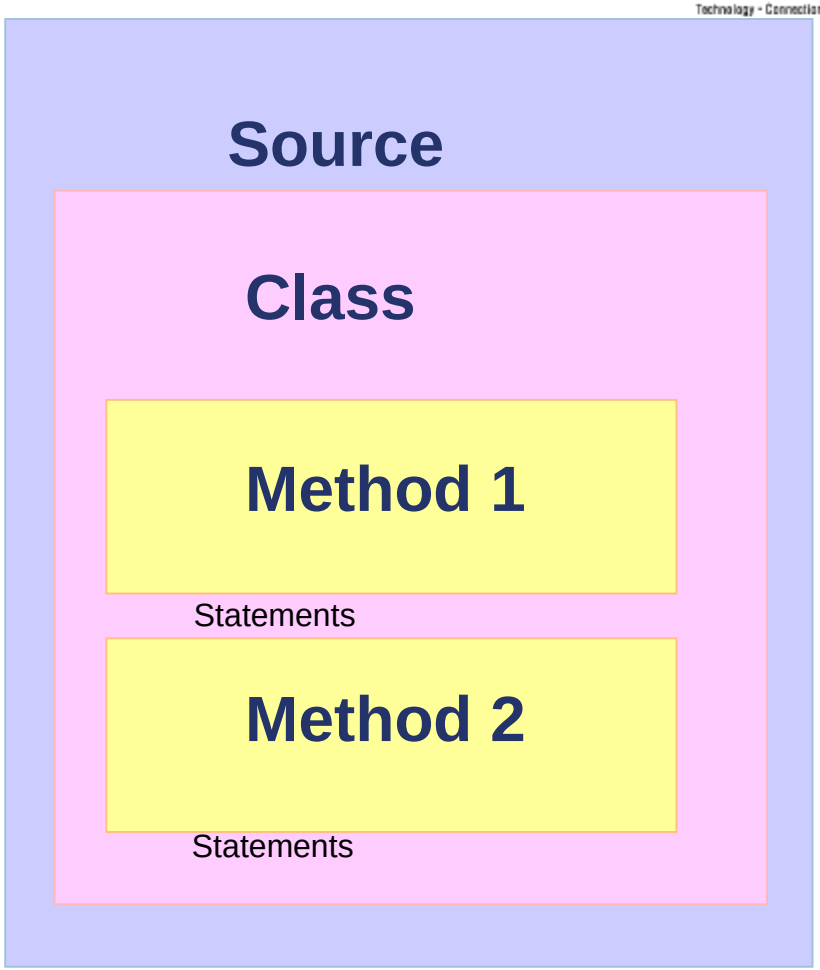
- Application Programming Interfaces (APIs)
  - Provides the core functionality of the Java programming language
  - A set of **class** libraries
    - From basic objects, to networking and security, XML generation and database access
    - Programmers uses when writing Java source code
- Included in 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

API Packages

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

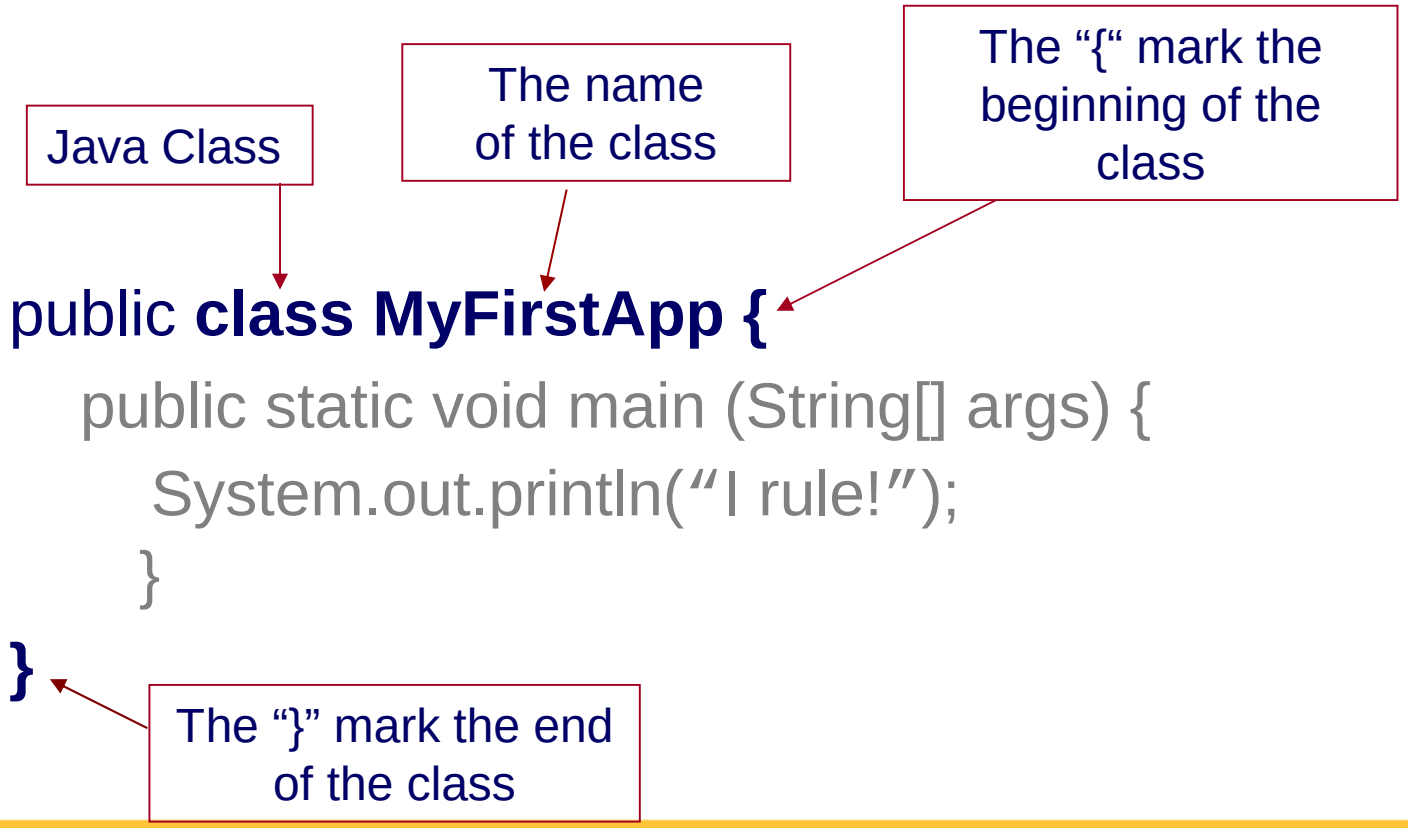
Java Code Structure

- 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***

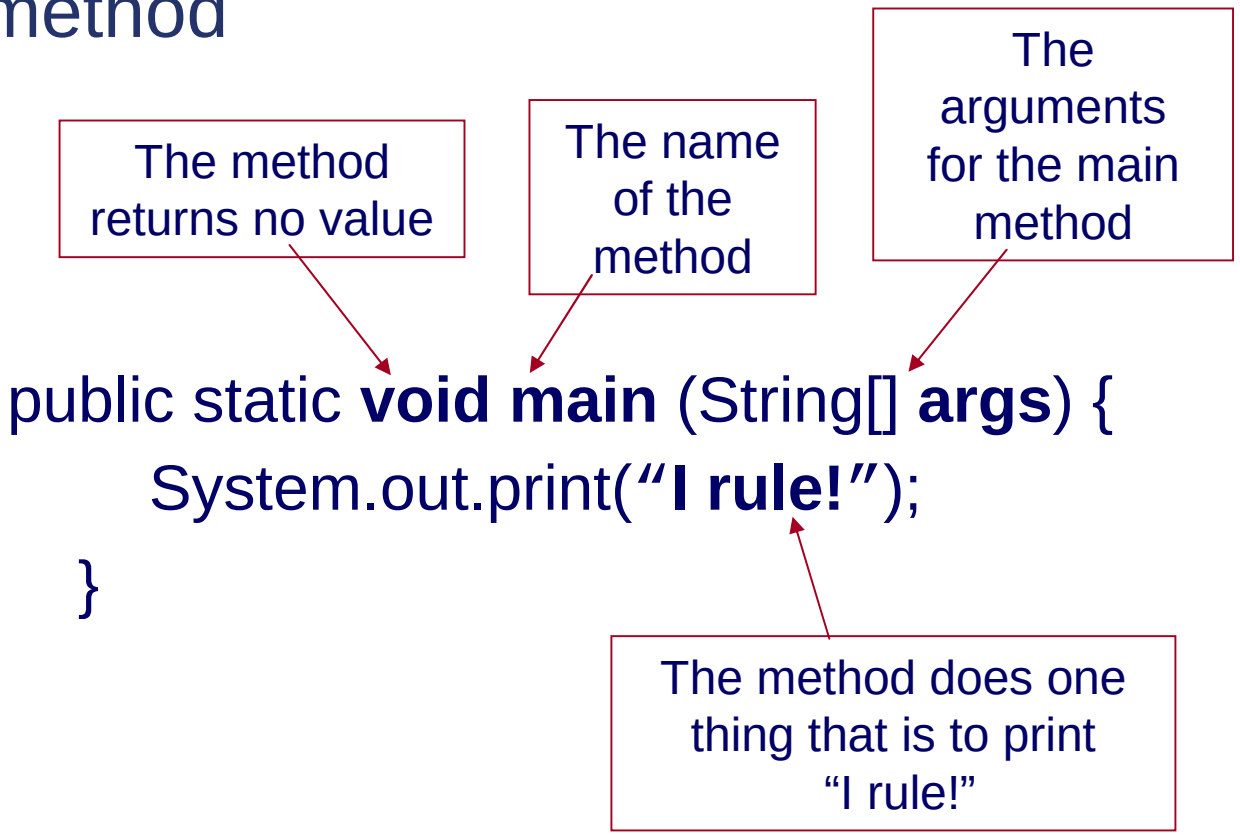


Java Class

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

Please see the crib sheet included in the hands-on lab instructions document

Classloaders and classpath

- Classloaders
  - Bootstrap - classes from core Java APIs
  - Extensions - classes that are standard extensions packages in the extensions directory
  - Application - classes from the local file system and it will load your application from the CLASSPATH

- classpath
  - A user defined environment variable used by Java runtime to determine where predefined/user-defined classes are located
  - User-defined classes that are complied with the javac compiler
    - i.e. command-line MyProgram.java → MyProgram.class

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.

Anatomy of an Array

```
int[] nums;
```

Declares an array of int's named "nums"

```
nums = new int[3];
```

Gives the Array object a length of [3]

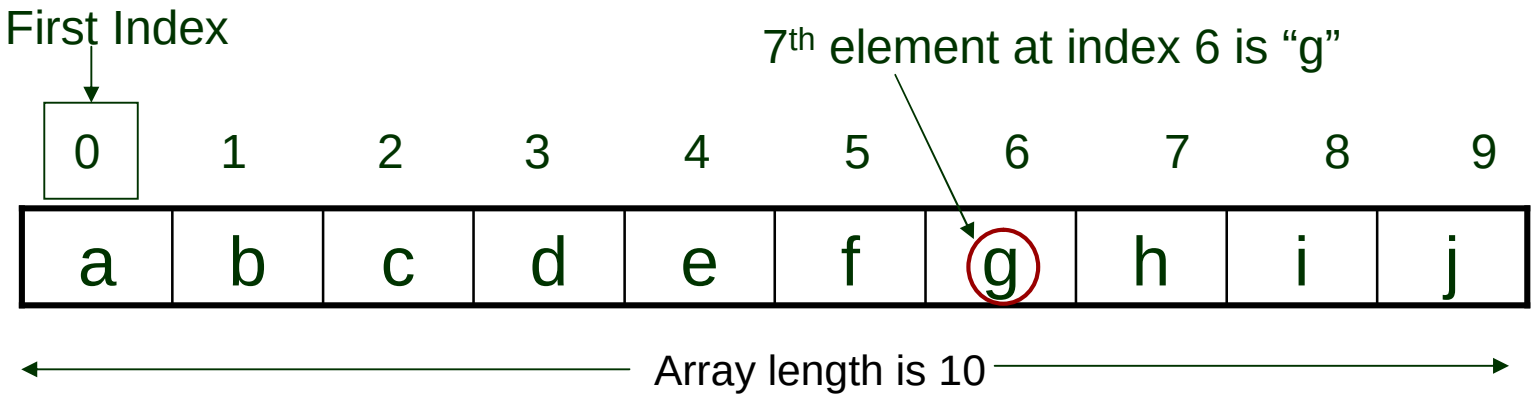
Instantiates an Array object with the key word "new"

```
nums[0] = 1;
nums[1] = 2;
nums[2] = 3;
```

Gives each element a value

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**]**;



Sample Array

```
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 … 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';
}
```

Break

- Like continue, but abandons entire loop instead of current iteration
- Can also use labels on break statements
- The break statement has two forms
  - Labeled and unlabeled
  - You can also use an unlabeled break to terminate a for, while, or do–while loop

```
for ( int i = 0; i < array.length; i++ ) {
    if ( array[ i ] == 0 ) {
        break;  // stop processing at first zero entry
    }
    // process element...
}
```

```
first:
  for ( int i = 0; i < array.length; i++ ) {
      if ( array[ i ] == 0 ) {
          break first;
          }
      // process element...
  }
```

For loops example

```
int i;
for (i=0 ; i < 10 ; i++) {
  System.out.println("i = " + i);
}
```

- i = 0
- i = 1
- i = 2
- ...
- i = 9

Common short hand:

```
for (int i=0 ; i < 10 ; i++) {
System.out.println("i = " + i);
}
```

Note: you can skip the " int I; "

While Loops

```
while ( boolean_expr ) {
  stmnts;
}
```

- expr evaluated at top of each loop
- body executed if expr evaluates to true
- Make sure your loop terminates!

```
int i = 0;

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

➔ i = 0
➔ i = 1
➔ i = 2
➔ ...
➔ i = 9

do .. while Loops

```
do {
    stmnts;
} while ( boolean_expr );
```

```
int i = 0;

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

- body executed each time through the loop
- boolean_expr is evaluated at the end of the loop
- body of the loop is always executed at least once

➔i = 0
➔i = 1
➔i = 2
➔...
➔i = 9

What is Eclipse?

- Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle
- Four download packages
  - Java IDE - If you are a Java developer
  - Java EE - If you are a Java developer creating Java EE application
  - C/C++ IDE – if you are a C/C++ developer
  - RPC - If you are planning to build Eclipse plugins and/or RPC applications

Eclipse Basics

- The Workbench
  - *Workbench* refers to the desktop development environment
  - Each Workbench window contains one or more *Perspectives*
  - More than one Workbench window can exist on the desktop at any given time
- Perspectives
  - Contain views and editors
    - Menus and tool bars
- Plug-ins
  - Eclipse based product is structured as a collection of *plug-ins*
  - Each plug-in contains the code that provides some of the product's functionality
- Software Updates
  - Under “Help” view pull down

Workbench Basics

- Resources
  - Projects
  - Files
  - Folders
- Import Wizard
  - Importing from the local file system
    - Menu bar File → Import
  - Importing existing projects
  - Importing resources from a zip file
- Export Wizard
  - Exporting to the file system
  - Exporting to an archive file
- Editors
  - More than one editor can be opened at the same time but only one can be active
- Console View
  - Running your Java Application
  - Messages will be displayed in the Console View
  - Problem View – syntax error, warnings…

About Eclipse Tooling

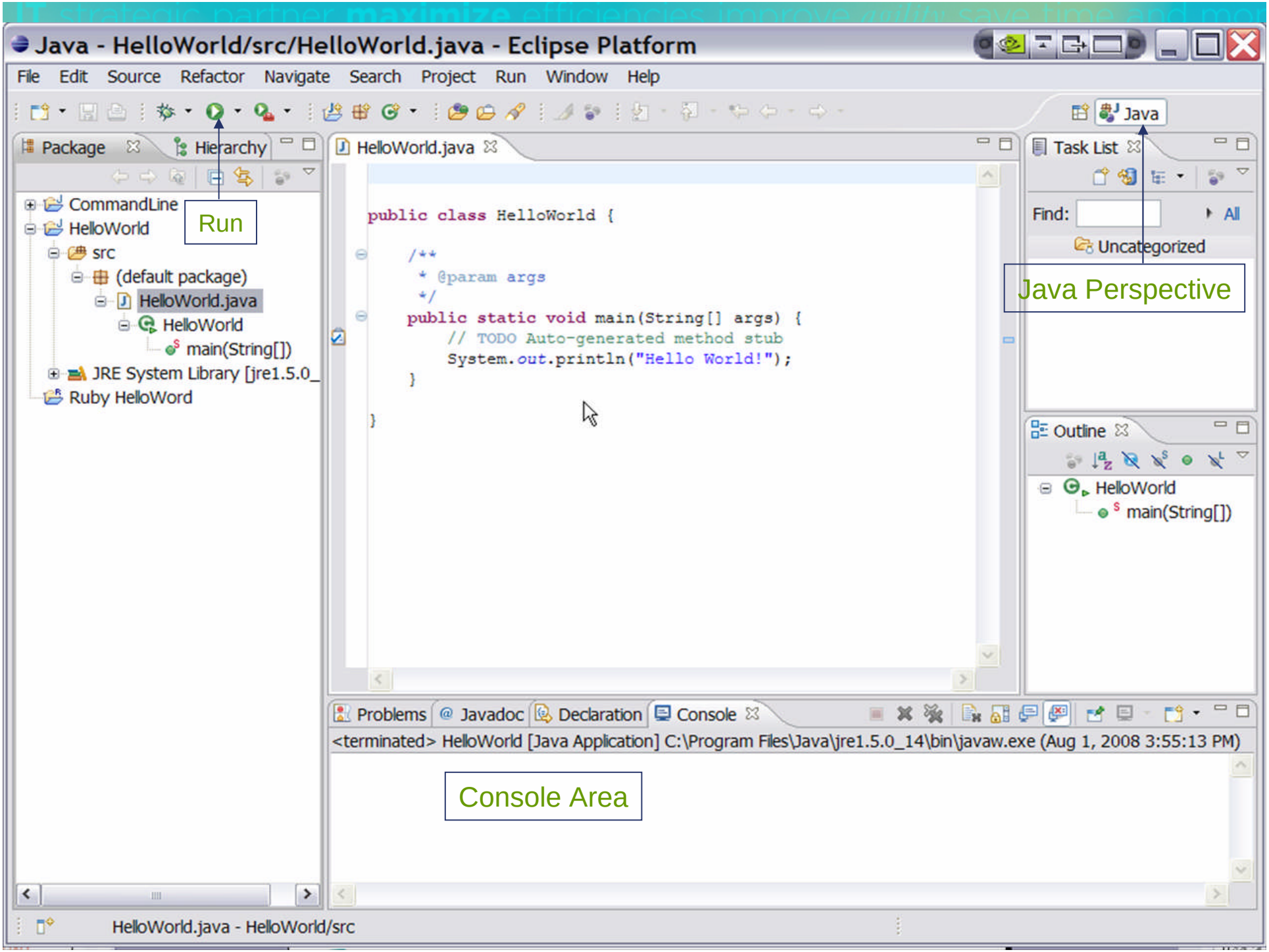
- Free download from http://www.eclipse.org
  - Click on Download Eclipse
  - The Java for the Beginner Labs use the Eclipse IDE for Java Developers
    - 

  - The Java EE Developer
    - 

  - 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

Eclipse Workspace at a Glance

- The Java and Resource Perspective
  - Editing, and syntax checking
  - Automatic code completion, identifying errors
  - Executing programs
- The Debug Perspective (session 1182 Lab II)
  - Local Debugging
  - Remote Debugging
- About testing with JUnit
  - Testing is an integral part of development
- Javadoc
  - Tool for generating documentation
    - doc comments in source code
    - HTML format
  - Java API documentation
    - http://java.sun.com/j2se/javadoc/


Java - HelloWorld/src/HelloWorld.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package
Hierarchy
CommandLine
HelloWorld
src
(default package)
HelloWorld.java
HelloWorld
main(String[])
JRE System Library [jre1.5.0_
Ruby HelloWord
Run
HelloWorld.java
public class HelloWorld {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Hello World!");
}
}
Task List
Find:
All
Uncategorized
Java Perspective
Outline
HelloWorld
main(String[])
Problems
Javadoc
Declaration
Console
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.5.0_14\bin\javaw.exe (Aug 1, 2008 3:55:13 PM)
Console Area
HelloWorld.java - HelloWorld/src

Lab Exercises

- Labeled Loop
- Do While Loop
- Command-Line
  - SumAverage

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

Sample Solution: Labeled Loop w/break Statement

```
public class LoopLabel {
public static void main (String arguments[]) {

myloop:
    for (int i = 1; i <= 6; i++)
    for (int j = 1; j <= 4; j++) {
        System.out.println("i is " + i + ", j is " + j);
          if (( i + j) > 5)
          break myloop;
                    }
          System.out.println("End of loops");
    }

}
```

Exercise #2: do...while

```
class DoWhileLoop {
      Public static void main (String arguments[]) {
            int a = 1;
            do {
                  System.out.println(“Looping, round  # “ + a);
                   a++;
            }
            while (a <= 10);
       }
}
```

```
class DoWhileLoop {
            public static void main (String arguments[]) {
                int a = 1;
                do { ……. }
                while (a <= 10);
                    System.out.println("Existing Do While Loop!");
                }
           }
```

Exercise #3: Command-Line SumAverage


```
public class SumAverage {
        public static void main(String arguments[]) {
                int sum = 0;

                for (int i = 0; i  < arguments.length; i++) {
                        sum += Integer.parseInt(arguments[i]);
                }

                System.out.println("Sum is: " + sum);
                System.out.println("Average is: " +
                    (float)sum / arguments.length);
        }
}
```

Thank You!

The Future Runs on System z