8352
Java from the Very Beginning, Part I

Steve Ryder
JSR Systems, Austin, Texas
Stephen Pipes
IBM Hursley Park Labs, United Kingdom

Reminder

- No food or drink in the lab (except for the speakers)
- Switch off mobile phones & pagers
- Don't hesitate to ask questions
- Have fun!

Objectives

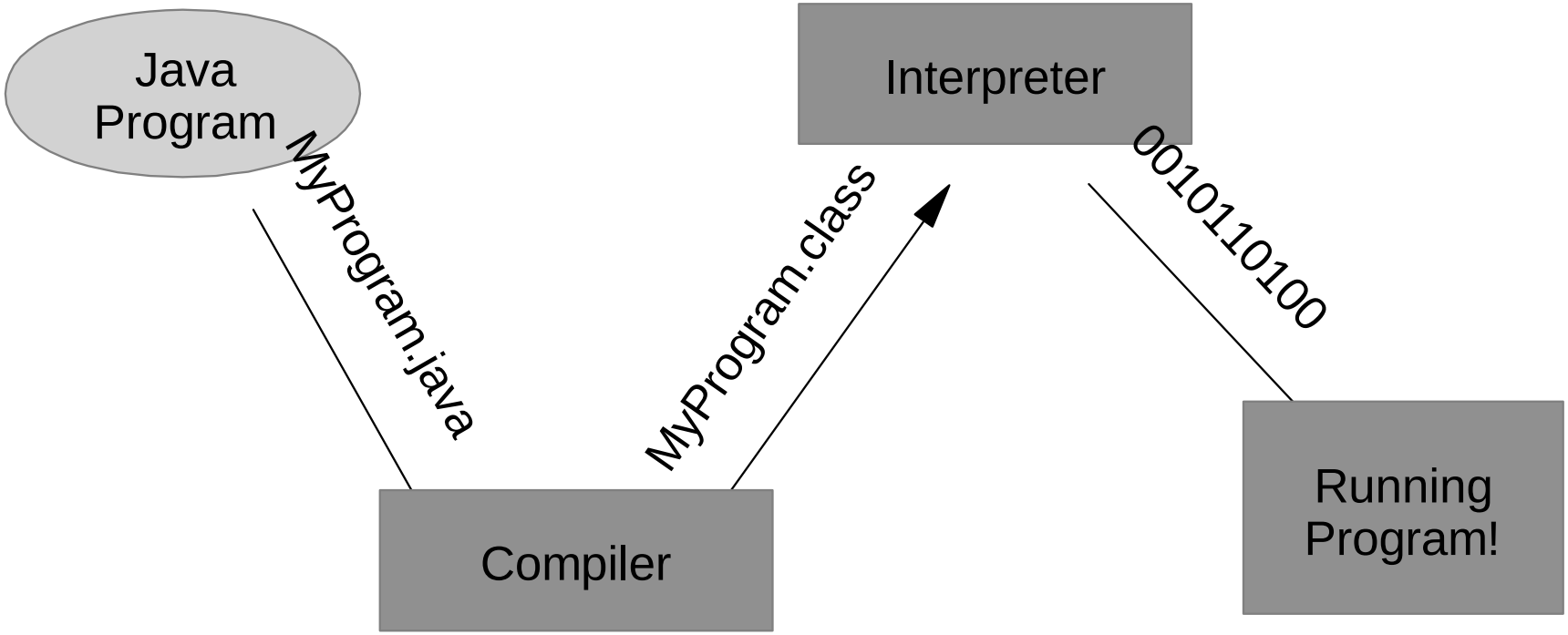
- Understand what Java is
- Understand what Java can do
- Be able to write simple Java programs

Agenda

- What is Java?
- Hello World
- Variables and Data types
- Operators
- Expressions
- Arrays and Strings

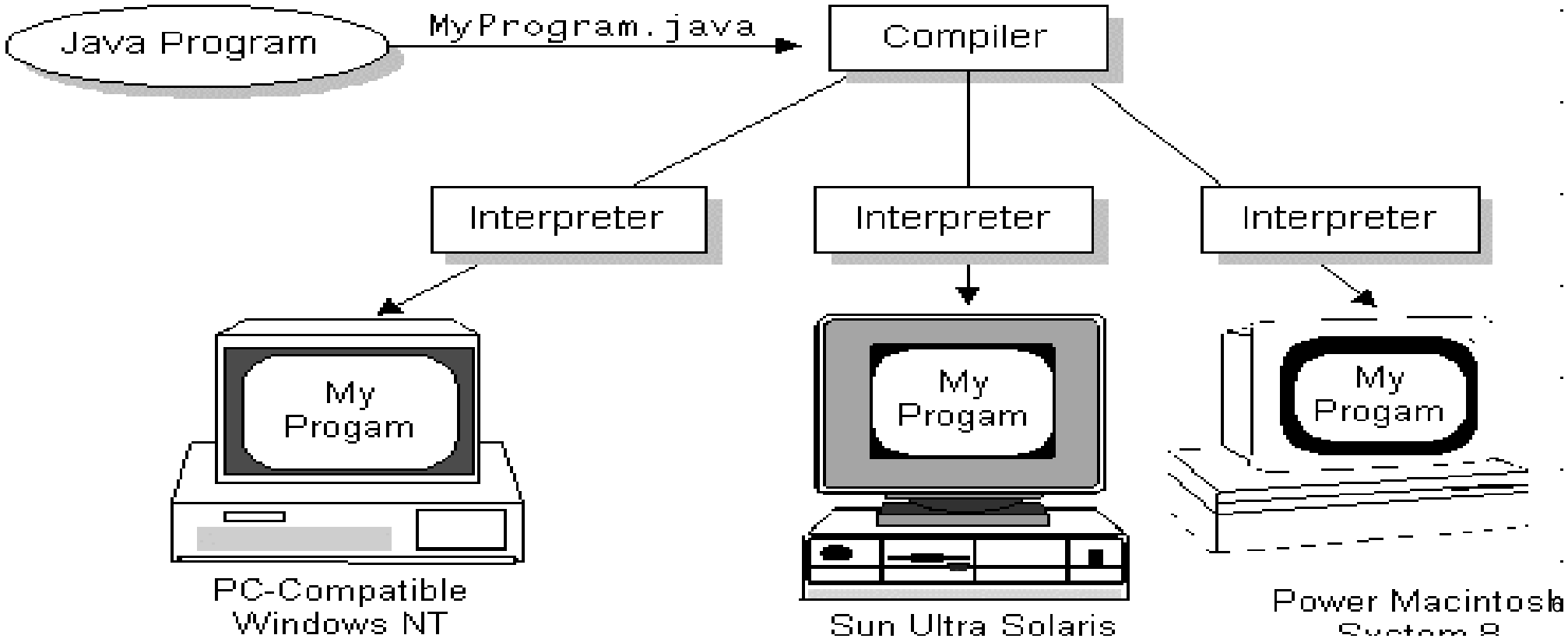
What is Java?

- A programming language
  - Compiled *and* interpreted



Java Bytecodes

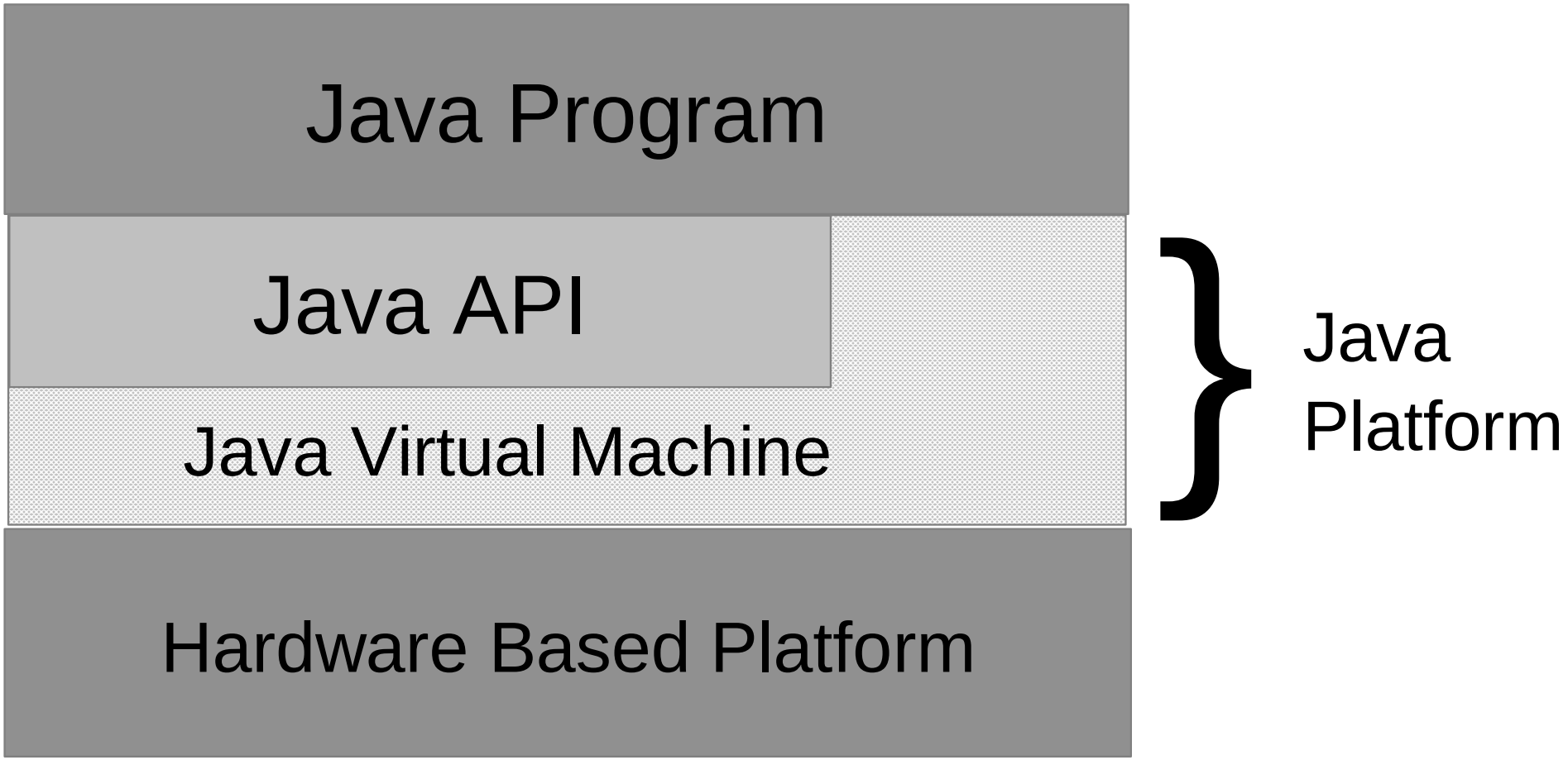
- "Machine language" for the Java Virtual Machine
- Enable "Write-once, run-anywhere"



What is Java?

- A platform
  - A software only platform that runs on top of other, hardware, platforms
  - Two components:
    - The Java Virtual Machine
    - The Java Application Programming Interface (API)

The Java Platform



What can Java Do?

- Applets
- Applications
- Beans
- Servers
- Servlets

The Core API

- Part of every full implementation of the Java platform
  - **The Essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
  - **Applets:** The set of conventions used by Java applets.
  - **Networking:** URLs, TCP and UDP sockets, and IP addresses.
  - **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
  - **Security:** Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.

The Core API

- **Software components:** Known as JavaBeans, can plug into existing component architectures such CORBA and Microsoft's OLE/COM/Active-X architecture.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC):** Provides uniform access to a wide range of relational databases.

Benefits of Java

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

Agenda

- What is Java?
- ➡ Hello World
- Variables and Data types
- Operators
- Expressions
- Arrays and Strings

Hello World

```
/**
 * 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.
    }
}
```

Compile the source file

- **javac** HelloWorldApp.java
  - produces a Java bytecode file called HelloWorldApp.class

Run the Interpreter

- **java** HelloWorldApp
  - Hello World!

Exercise

- Create "Hello World" program
- Compile it
- Run it

Common Compiler Problems

- Can't find the compiler
  - javac: Command not found
  - modify your PATH to include the Java tools directory
- Syntax errors
  - type of error, line number, code on line, position of error

```
testing.java:14: `;' expected.
System.out.println("Input has " + count + " chars.")
                                                    ^
1 error
```

  - edit your source code and try again

Common Compiler Problems

- Semantic errors
  - Basic correctness checks, e.g. use of a variable before it has been initialised

    testing.java:13: Variable count may not have been initialized.
    count++;
    ^

    1 error
  - Edit your source and try again

Common Interpreter Problems

- Can't find class
  - Can't find class HelloWorldApp.class
  - the argument to the interpreter is the name of the class you want to use (HelloWorldApp), not the filename (HelloWorldApp.class)
- The main method is not defined
  - In class HelloWorldApp: void main(String argv[]) is not defined
  - The interpreter must have a main method to know where to begin executing

Hello World Dissected

```
/**
 * 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.
    }
}
```

Comments

- /* text */
  - the 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

Class Definition

- The skeleton of any Java program

```
class name {
   ...
}
```

All variables and methods of a class are placed between the curly braces. HelloWorldApp has no variables and one method, main.

➡ Java has no global methods (functions) or data (variables)

The main method

```
class HelloWorldApp {
    public static void main(String[ ] args) {
      System.out.println("Hello World!"); //Display the string.
    }
}
```

- Every Java application must contain a **main** method with signature:
  public static void main(String[ ] args)
  - public - the method may be called by any object
  - static - indicates that this is a class method
  - void - the method returns no value
  - String [] args - the method accepts a single argument, which is an array of strings

Agenda

- What is Java?
- Hello World
- ➡ Variables and Data types
- Operators
- Expressions
- Arrays and Strings

Variables and Data Types

- A variable declaration always contains two components: the *type* of the variable and its name. The location of the variable declaration, that is, where the declaration appears in relation to other code elements, determines its *scope*.
- Type
  - determines values a variable can contain
  - AND, the operations that can be performed on it

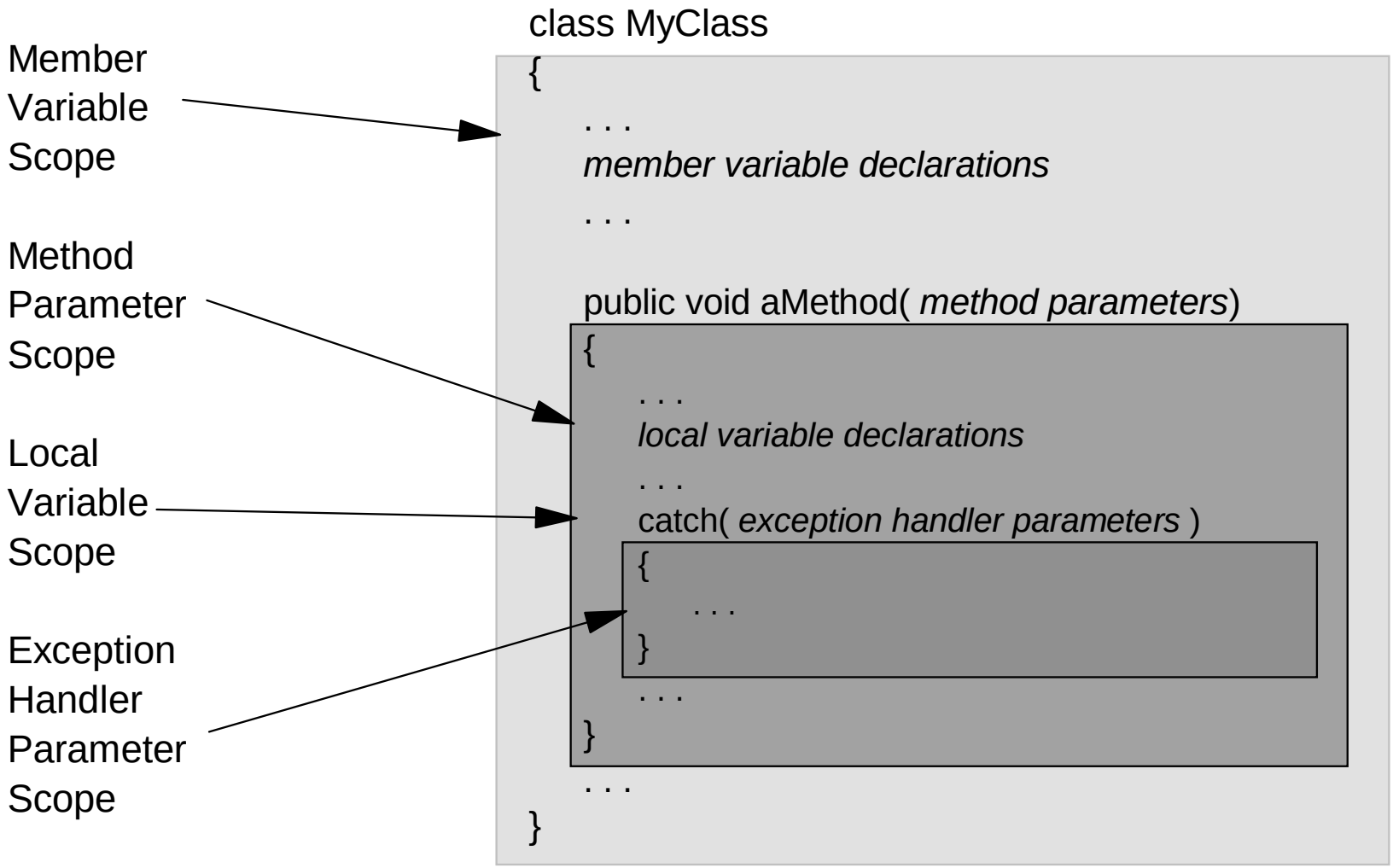
Primitive Types

- byte 8 bit integer $-2^7..2^7-1$ 10
- short 16 bit integer $-2^{15}..2^{15}-1$ 15
- int 32 bit integer $-2^{31}..2^{31}-1$ 9
- long 64 bit integer $-2^{63}..2^{63}-1$ 18L
- float 32 bit float 1.0f
- double 64 bit double-precision float 1.0d
- char character literal 'a'
- boolean boolean value true or false

Variable Names

- Begin with letter, dollar sign ($) or underscore (_)
- Followed by letters, underscores, dollar signs or digits
- Convention: variable names begin with a lower case letter, if the name contains multiple words, the words are joined together and each subsequent word begins with a capital letter.
  - ✓ isUpper

Scope


Member
Variable
Scope
Method
Parameter
Scope
Local
Variable
Scope
Exception
Handler
Parameter
Scope
class MyClass
{
. . .
member variable declarations
. . .
public void aMethod(method parameters)
{
. . .
local variable declarations
. . .
catch(exception handler parameters)
{
. . .
}
. . .
}
. . .
}

Reference Types

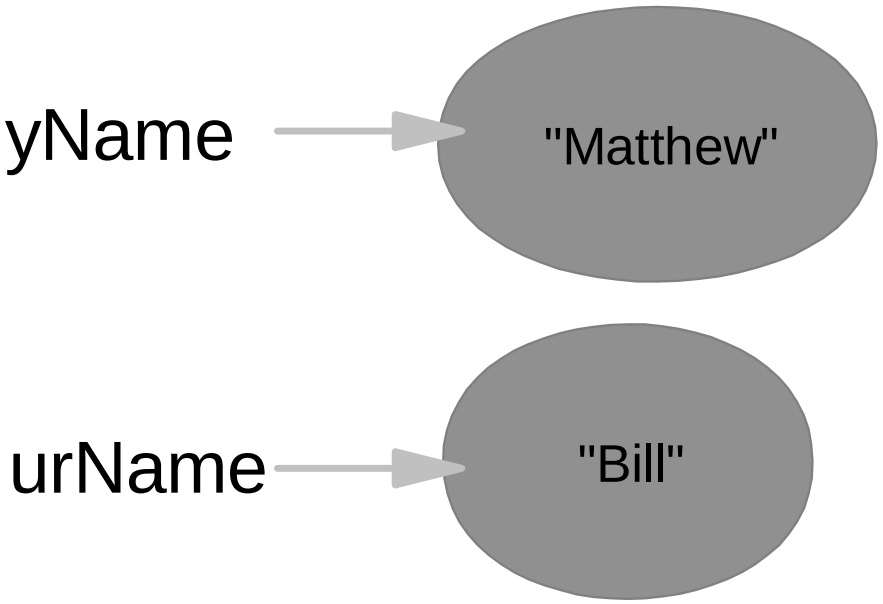
- Anything that is not a primitive type is a *reference* type
  - arrays
  - classes
  - objects
  - interfaces
- The value of a reference variable is a reference to the actual value or set of values represented by that variable

Reference Types

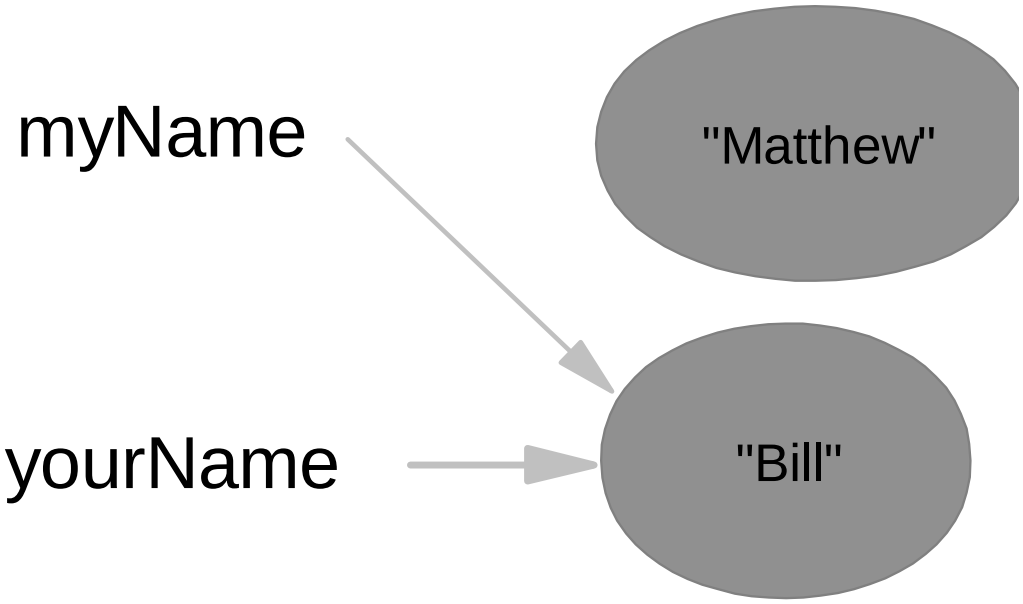
```
{
  String myName;    // myName is a reference to a String object
  String yourName;  // yourName is a reference to a String object

  myName = "Matthew";
  yourName = "Bill";

  myName = yourName;  // both variables refer to the SAME
                                    // String object
}
```

Reference Types

efore assignment



After assignment



Agenda

- What is Java?
- Hello World
- Variables and Data types
- ➡ Operators
- Expressions
- Arrays and Strings

Operators

- +,-,/,*,%
  - a = b + c;
  - For integer types, "/" is integer division, "%" is remainder
  - For floating point, "/" is floating point division
- ++,--
  - ++foo adds 1 to foo, and uses the result in expr
  - foo++ uses foo in expr, and then adds 1!

```
int foo = 5;
System.out.println( ++foo );   // prints 6
System.out.println( foo++ );   // prints 6
System.out.println( foo );     // prints 7
```

Operators

- >,>=,<,<=,==,!=
  - if (a >= b)
  - Watch out!! == tests for equality, = is the assignment operator!
- &&, ||, !
  - if (a && b)
- >>,<<,>>>,&,|,^,~
  - >>, << signed shift by n bits (192 >> 1 == 96)
  - >>> is an unsigned right shift
  - &,| are bitwise AND, OR
  - ^ is a bitwise exclusive OR
  - ~ is a bitwise inversion

Assignment Operators

- =, +=, -=, *=, /= , %=
  - a += b; is equivalent to: a = a + b;
  - a -= b; is equivalent to: a = a - b;
  - a *= b; is equivalent to a = a * b;
  - a /= b; is equivalent to a = a / b;
  - a %= b; is equivalent to a = a % b;

Agenda

- What is Java?
- Hello World
- Variables and Data types
- Operators
- ➡ Expressions
- Arrays and Strings

Expressions

- An expression is a series of variables, operations and method calls that evaluate to a single value
- Use parenthesis to specify precedence order

x + (y / 100)

Exercise

- Write a program to convert the distance of a marathon in miles and yards to kilometres.

Marathon

- 41.834
- 42.18596875
- Something else?

Marathon

```
public static void main(String[] args) {
    int miles;
    int yards;
    double kilometers;

    // Set miles to 26, and yards to 385
    miles = 26;
    yards = 385;

    // One mile is 1.609 kilometers
    // There are 1760.0 yards in a mile
    kilometers = 1.609 * (miles + (yards / 1760.0));

    // Print the answer
    System.out.println("A marathon is " + kilometers + " kilometers.");
}
```

Agenda

- What is Java?
- Hello World
- Variables and Data types
- Operators
- Expressions
- ➡ Arrays and Strings

Arrays

- Declare as *type*[] varName;
  - **int[ ]** myInts;
- Must allocate memory before use:
  - myInts = **new** int[10];

- General form:
  - **elementType[ ]** arrayName = **new elementType[** arraySize **]**;

Arrays

- Array indices always start at zero.
- Access array elements using []:
  - myArray[0] = 5;
- Special array property *length*
  - myArray.length

Arrays

- **int [ ]** squares = **new int[**5**]**;
-
- squares[0] = 0;
- squares[1] = 1;
- squares[2] = 4;
- squares[3] = 9;
- squares[4] = 16;

Strings

- "String literal"
- Implemented by String class
  - String name = "Adrian";
- Immutable
- Concatenate with + operator
  - "Counted " + count + " chars."

Summary

- Java is a programming language and a platform
- edit-compile-interpret cycle
- All applications need a "main" method
- Types are platform independent
- C-style syntax