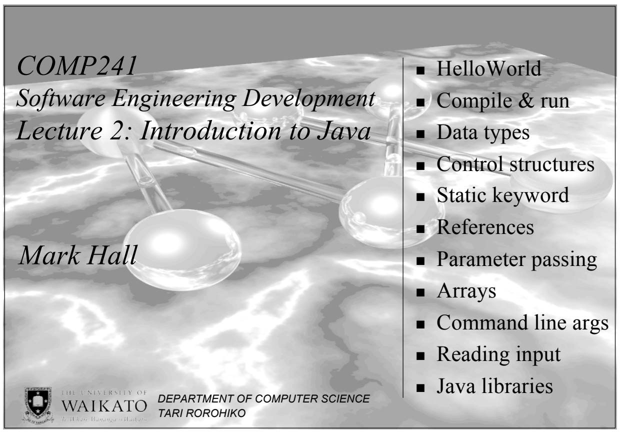


COMP241

Software Engineering Development

Lecture 2: Introduction to Java

Mark Hall

- HelloWorld
- Compile & run
- Data types
- Control structures
- Static keyword
- References
- Parameter passing
- Arrays
- Command line args
- Reading input
- Java libraries

THE UNIVERSITY OF WAIKATO

DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Reading

- Text book chapter 1: *A Crash Course in Java.*
- Bruce Eckel. *Thinking in Java*, Prentice-Hall, 2002. `http://www.mindview.net/Books/TIJ/`
- These links are also on our course homepage. `http://www.cs.waikato.ac.nz/~mhall/241`

Our 1st Java Program

“Hello-World” in Java:

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Compiling and Running

Setup

- `java-setup`
- `javac-setup`

Compile your program

- `javac HelloWorld.java`

Run your program

- `java HelloWorld`
- *Note:* Java is interpreted

The Class Name

The class name is **important**:

- `public class HelloWorld {...}`
- Handle for object creation
- Filenames must match class names:
  - Source Filename: `HelloWorld.java`
  - Compiled File: `HelloWorld.class`

Data Types in Java

Possible Declarations:

- Primitive types: `int`, `char`, `double`, …
  - predefined, cannot define ourselves
  - names start with lower case letters
- Classes
  - built-in classes, e.g. `String`
  - user-defined classes
  - names start with capital letter by convention

Primitive Data Types

Primitive Type	Size	Minimum	Maximum
boolean	–	false	true
char	16 bit	Unicode 0	Unicode 65535
byte	8 bit	-128	127
short	16 bit	-32768	32767
int	32 bit	-2^{31}	$2^{31}-1$
long	64 bit	-2^{63}	$2^{63}-1$
float	32 bit	IEEE 754	IEEE 754
double	64 bit	IEEE 754	IEEE 754

Primitive Type `boolean`

- Java has `boolean` as a primitive type.
- Possible values are:

```
true
false
```

- Booleans cannot be assigned or compared to numeric types!

Strings

- There is a built-in type `String`
- Many functions:
  - '+' for concatenation
  - '`length()`' method to count characters
  - '`toString()`' methods for conversion
  - '`substring()`' to extract part of a `String`

Control Structures

- Statements very similar to C/C++/C#
  - Assignments
  - `if`, `for`, `while`, `switch`, …
  - `return`
- Expressions also similar to C/C++/C#
  - Type cast using `(char)10` style notation
- There is no `struct`
  - Use `class` instead

The `static` Keyword

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

- *Static* indicates a method that is called independently of objects.
- Static methods work like traditional functions in C/C++.
- This is an unusual case in Java.

Object-Oriented "Hello World"

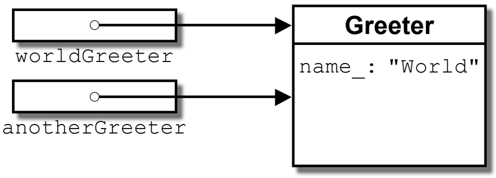
```
public class Greeter {
  public Greeter(String name) {
    name_ = name;
  }
  public void sayHello() {
    System.out.println
          ("Hello, " + name_ + "!");
  }
  private String name_;
}
```

Object-Oriented "Hello World" ***ctd.***

```
public class HelloWorld {
  public static void main(String[] args){
    Greeter worldGreeter =
           new Greeter("World");
    worldGreeter.sayHello();
  }
}
```

References

Java Objects are implemented as ***references****, i.e., pointers.*



```
Greeter worldGreeter = new Greeter("World");
Greeter anotherGreeter = worldGreeter;
```

Parameter Passing

- Parameters are passed "**by-value**".
- Copied before passing into methods.

```
public class Greeter {
  /**
   * Tries to store the length of this greeter's
   * name in an integer variable.
   */
  public void storeLength(int n) {
    n = name_.length();
  }
}
```

Object Parameter Passing

- Object parameters are passed by copying the ***reference***, not the object.

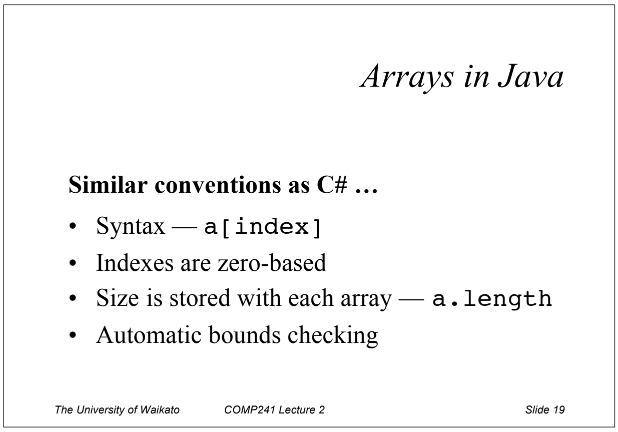
```
public class Greeter {
  /**
   * Sets another greeter's name
   * to this greeter's name.
   */
  public void putName(Greeter other) {
    other.name_ = name_;
  }
}
```

What does this Code Produce?

```
Greeter worldGreeter =
          new Greeter("World");
Greeter robiGreeter =
          new Greeter("Robi");
worldGreeter.putName
          (robiGreeter);
worldGreeter.sayHello();
robiGreeter.sayHello();
```

What does this Code Produce?

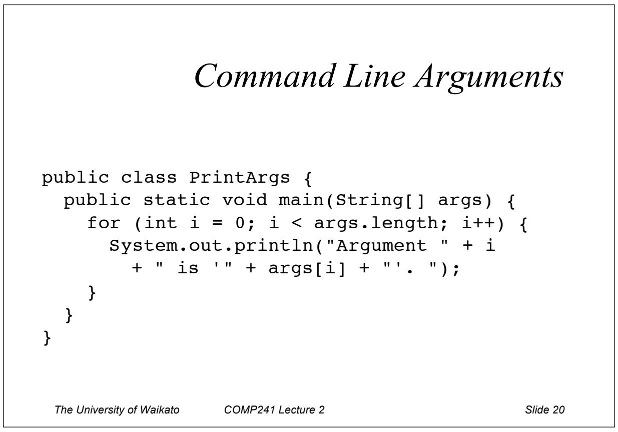
```
Greeter greeter1 =
          new Greeter("World");
Greeter greeter2 =
          new Greeter("Robi");
Greeter greeter3 = greeter2;
greeter1.putName(greeter2);
greeter1.sayHello();
greeter2.sayHello();
greeter3.sayHello();
```



Arrays in Java

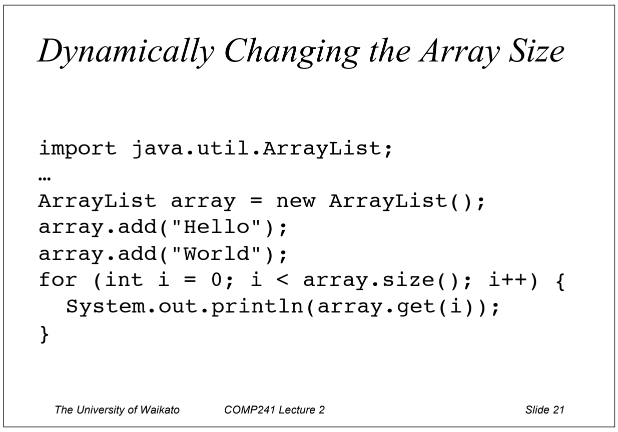
Similar conventions as C# …

- Syntax — `a[index]`
- Indexes are zero-based
- Size is stored with each array — `a.length`
- Automatic bounds checking



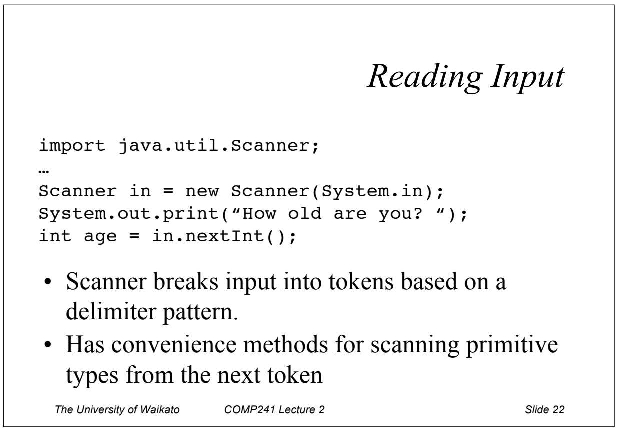
Command Line Arguments

```
public class PrintArgs {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
      System.out.println("Argument " + i
        + " is '" + args[i] + "'. ");
    }
  }
}
```



Dynamically Changing the Array Size

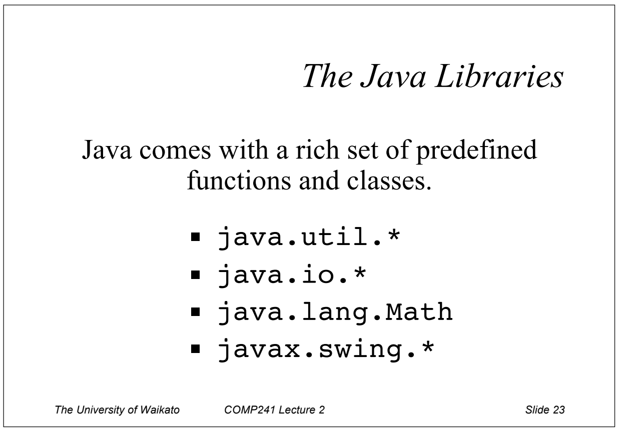
```
import java.util.ArrayList;
…
ArrayList array = new ArrayList();
array.add("Hello");
array.add("World");
for (int i = 0; i < array.size(); i++) {
  System.out.println(array.get(i));
}
```



Reading Input

```
import java.util.Scanner;
…
Scanner in = new Scanner(System.in);
System.out.print(“How old are you? “);
int age = in.nextInt();
```

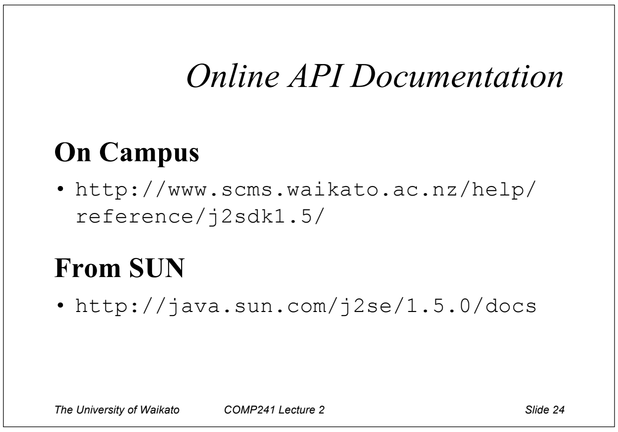
- Scanner breaks input into tokens based on a delimiter pattern.
- Has convenience methods for scanning primitive types from the next token



The Java Libraries

Java comes with a rich set of predefined functions and classes.

- `java.util.*`
- `java.io.*`
- `java.lang.Math`
- `javax.swing.*`



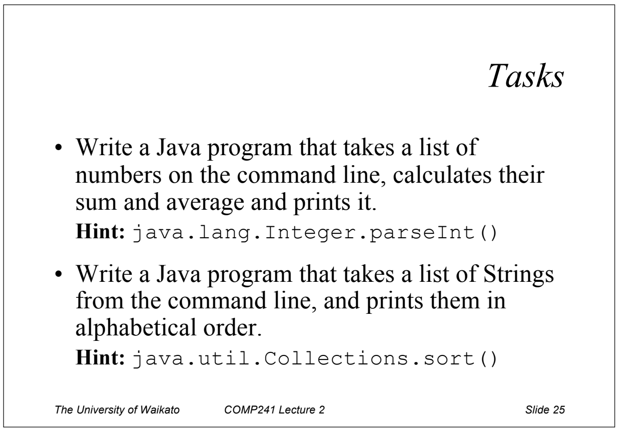
Online API Documentation

On Campus

- `http://www.scms.waikato.ac.nz/help/reference/j2sdk1.5/`

From SUN

- `http://java.sun.com/j2se/1.5.0/docs`



Tasks

- Write a Java program that takes a list of numbers on the command line, calculates their sum and average and prints it.
  Hint: `java.lang.Integer.parseInt()`
- Write a Java program that takes a list of Strings from the command line, and prints them in alphabetical order.
  Hint: `java.util.Collections.sort()`