Lancaster University

SCC.110 Software Development
Java Syntax Reference

Prof Joe Finney and Dr Kelly Widdicks
j.finney@lancaster.ac.uk, k.v.widdicks@lancaster.ac.uk

Java syntax reference

Lancaster University

- As we have seen, Java has a very similar syntax to C… but there are some subtle differences
- Read the slides and examples in this handout to learn how to translate what you have learned in C into Java
- You will need some of this knowledge to complete this week's lab exercise
- If there are any concepts you do not understand, ask one of your TAs about it

Java primitive types 1

Lancaster University

- **byte**: 8 bit signed integer
- **short**: 16 bit signed integer
- **int**: 32 bit signed integer
- **long**: 64 bit signed integer

```
byte b = 127;       // decimal by default
short s = 32767;
int i = 2147483647;
long l = 9223372036854775807;
long l = 0xCAFE; // hexadecimal (for hackers only!)
```

- **float**: 32 bit floating point number
- **double**: 64 bit floating point number

```
double d = 123.72 + 1.22;  // can have fractional parts
float f = 123.72f;  // double is assumed by default

float f = (float) d;// numeric types can be casted
int i = (int) f;        // integer casts round down!
```

Java primitive types 2

Lancaster University

- **boolean**: true or false

```
boolean isHere = true;              // booleans can be assigned
boolean isLearning = false;
boolean isBored = ! isLearning          // negated
boolean isGoingToFail = !(isHere && isLearning)      // evaluated with && and ||
```

- **char**: unicode character

```
char c = 'j' ;        // single quotes only!
```

- **void**: no type!

Java variables

Lancaster University

- Unlike C, Java variables can be defined **anywhere**! They do not need to be declared at the start of a method
- They can even be declared in middle of a code block, but they are only in scope as defined by the enclosing code block

```
public class HelloWorld {
        int a = 0;
        public static void main( String[ ] arguments ) {
                System.out.println( "Hello! " );
                int b = 0;
                while (true) {
                        System.out.println( "Let me out!!!" );
                        b++;
                }
        }
}
```

Java comparisons 1

Lancaster University

- If statements also work like they do in C
- Tests can be done with the usual operators (== != < > <= >=)

```
public class HelloWorld {
        int a = 0;
        public static void main( String[ ] arguments ) {
                int a = 10;
                int b = 20;
                if (a > b)
                        System.out.println( "Yay!!" );
                else
                        System.out.println( "Boo!!" );
        }
}
```

Java comparisons 2

Lancaster University

- Tests themselves now evaluate to a **boolean** type!

```
public class HelloWorld {
        int a = 0;
        public static void main( String[ ] arguments ) {
                int a = 10;
                int b = 20;
                boolean isBigger = a > b;
                if (isBigger == true)
                        System.out.println( "Yay!!" );
                else
                        System.out.println( "Boo!!" );
                if (isBigger)
                        System.out.println( "Yay again!!" );
        }
}
```

Java comparisons 3

Lancaster University

- Take care though… what is wrong here?

```
public class HelloWorld {
        int a = 0;
        public static void main( String[ ] arguments ) {
                int a = 10;
                int b = 20;
                boolean isBigger = a > b;
                if (isBigger)
                        System.out.println( "Yay!!" );
                        System.out.println( "Bigger!" );
                else
                        System.out.println( "Boo!!" );
                        System.out.println( "Smaller!" );
        }
}
```

Java comparisons 4

Lancaster University

- That's better!

```
public class HelloWorld {
        int a = 0;
        public static void main( String[ ] arguments ) {
                int a = 10;
                int b = 20;
                boolean isBigger = a > b;
                if (isBigger) {
                        System.out.println( "Yay!!" );
                        System.out.println( "Bigger!" );
                }
                else {
                        System.out.println( "Boo!!" );
                        System.out.println( "Smaller!" );
                }
        }
}
```

Loops: while

Lancaster University

- While loops are identical to C
  - Syntax: while(boolean_condition) { code to execute }

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                int i = 0;
                while (i < 10) {
                        System.out.println( "Wheeeeeee!" );
                        i++;
                }
        }
}
```

Loops: for

Lancaster University

- For loops also work much the same as they do in C, except we can now create loop variables as we need them (which is safer!)
  - Syntax: for (init_statement; boolean_condition; modifier_statement)
  - **The loop variable is only in scope within the loop.**

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                for (int i = 0; i < 10; i++) {
                        System.out.println( "Wheeeeeee!" );
                }
        }
}
```

Strings 1

Lancaster University

- Strings in Java are very different to strings in C
  - C uses an array of characters to represent a string
  - Java has a specific String type that's safer and more expressive
- String has a capitalized first letter unlike primitive types
  - By convention, all classes are named in this way
  - String is actually a Java class, written in Java! ☺
- When copying and pasting, **be careful that the quotes are correct!**

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

Strings 2

Lancaster University

- Strings are not pointers!
  - Java doesn't have pointers… this avoids many complexities of C!
  - Strings **cannot** be accessed like arrays

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                String h = " Hello ";
                h[0] = 'C';
                System.out.println(h);
        }
}
```

Strings 3

Lancaster University

- Strings can be concatenated to form other strings
  - Uses the + operator
  - Much simpler than C printf syntax!

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                String h = "Hello";
                String w = "World";

                System.out.println(h + w);          //Outputs HelloWorld
                System.out.println(h + " " + w);    //Outputs Hello World
        }
}
```

Strings 4

Lancaster University

- Use the **length()** method to find out the length of a String

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                String name = "Joe";

                if (name.length() < 4)
                        System.out.println( "That's a short name!" );
        }
}
```

Strings 5

Lancaster University

- **Take care when comparing strings**
  - Do not use **==** and be very wary if you ever see **==** used with anything that is not a primitive type
  - Use the **equals()** method

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                String name = "Joe";
                String name2 = "Joe";

                if (name.equals(name2))
                        System.out.println("That's me!");
        }
}
```

Strings 6

Lancaster University

- Use Escape Sequences to represent reserved Java characters

character	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

```
String h = "Hello";
String w = "World";
System.out.println("\"" + h + " " + w + "\", said Joe");
```

Arrays

Lancaster University

- Arrays take the same form as C
  - Declared and accessed using square brackets [ ]
  - Are of fixed length, initialized using braces { }

```
public class HelloWorld {
        public static void main( String[ ] arguments ) {
                String[] weekDays = { "Mon", " Tue ", "Wed", "Thu", "Fri"};
                System.out.println( weekDays[0] );
        }
}
```

  - Except arrays in Java know how long they are using **.length**

```
for (int i = 0; i < arguments.length; i++)
        System.out.println(argument[i]);
```

Summary

Lancaster University

- Java is designed to be:
  - General purpose
  - Simple, safe and open
  - Object Oriented
  - Ideal for reuse and extensibility
  - Universal