NOTES ON JAVA FOR HADOOP

BIG DATA PROCESSING

Félix Cuadrado

felix.cuadrado@qmul.ac.uk

Queen Mary University of London

School of Electronic Engineering and Computer Science

The module is NOT about Java

- BUT, we have to use Java for developing Hadoop programs
- We will only use a very limited set of Java, in order to write our Map, Reduce, Combine functions, as well as the Job programs that tie all together
- We will use the same pattern over and over again

Java concepts that you should know for Hadoop

- Base Java syntax, .java structure
- Basic primitive types and operations (int, double, Boolean). Defining variables and using them
- Control structures: conditionals, loops
- Constructors for creating class instances
- Format of a class file: header, attributes, and methods
- Strings, text manipulation in Java
- Basic Java collections: using arrays, HashTable, List
- *Recommended setup for ITL Java Hadoop projects (lab 1)*
- *Defining Mapper, Reducer, Job classes (lab 1)*

Good Java learning resources

- http://introcs.cs.princeton.edu/java/home/
  - Chapters 1.1-1.4, 2.1,2.2, 3.1 coverthe main concepts we will use.
- I recommend combining with an Eclipse Java tutorial
  - https://www.youtube.com/watch?v=pKJMWpMKev8
- QMUL library also has many Java books
- Stackoverflow (with moderation, and understanding what does the solution do)

Java concepts that you don’t need to know for this module (but it’s great if you do)

- Creating your own classes: inheritance, polymorphism, abstract/interface concepts and details.
- Garbage collection
- Execution threads
- Details of handling input/output streams in Java
- Software Engineering (as in how to design a complex project)
- **All Hadoop projects follow the same structure we learn from lab 1**.
  - Copy//Paste//Rename works quite well to start a new project.

Java 101

- Java is a procedural programming language
  - When running a program, the runtime executes one line at a time, each one after the previous one
  - Control blocks modify the flow of the program
    - E.g. **`if`**, **`for`**, **`while`** …
  - Java programs begin by running a **`main`** method
- Java is a compiled language
  - .java text files are compiled into .class binary files with the same name
  - Automatically done by IDE, and the Ant build system

A Java project: classes

- The project is a collection of class definitions
  - Each class is stored in a .java source file
  - The name of the class MUST match exactly the file name
    - Use an IDE (like Eclipse) to make your life easy
- The project will also make use of external classes, either part of the base Java distribution (for example, a Class for accessing a File), or an external library

Java syntax

- Java is Case SenSitive
- Developers follow conventions to keep sanity
  - All Class Names Begin With Caps
  - All object names are written in lowercase
  - This is not needed, but eases reading
- The program has a set of blocks, all of them defined by opening { and closing } brackets

```
class Dog {
}
```

Java classes

- Classes have a first line with the name of the class
- A collection of attributes (variables for class objects)
- And a collection of methods
  - Methods can have a set of input parameters, and optionally return something
  - There is a special method called the constructor, it creates new objects from the class

Using an Object Oriented Language

- You create objects (instances of classes) with the new Constructor
  - Mapper map = new WordCountMapper();
  - Objects combine data structures with functions
- The methods you execute receive references to existing objects as method arguments
  - public void map(Context ctx)
- You invoke methods from objects using .
  - context.write(),   words.add("The")

Using objects in Java

- A Java program is composed by a series of objects, that use functionality from each other
- In order to use an object, we use a Reference for the object (similar to a variable in other languages)
- Example
  - Hashtable words = new Hashtable();
  - words.add("the",1)
  - //new is the constructor method, that creates a new object from a class definition. You can invoke methods from the object using the refrence
  - Method arguments provide existing references to objects

Java program execution

- Begins by running one main class method.
- The main method creates objects from classes, and invokes methods from these objects
- Each method might in turn create more objects, and invoke methods from them.
- Until the program ends.

Java jar files

- Java projects compile the .java classes into .class binary files, and package them in zip files with jar extension.
- Hadoop is distributed as a collection of jar files
- Requirements for Hadoop (for example one library that contains classes that create a server for worker communications)
- Each Hadoop project will be packaged into a jar file

A Hadoop MapReduce program

- We don’t start Hadoop (Hadoop is already running in our system/cluster).
- Instead, we submit a job, stored inside a jar file
  - hadoop jar dist/WordCount.jar WordCount input out
- Hadoop will create one instance of our job class. The job object will create instances of our designated Mapper, Reducer, and other
- Hadoop will invoke the run method of our job, that starts the job execution