**CSE470: SOFTWARE ENGINEERING**

**ASSIGNMENT01**

**LIBRARIES**

Here is a brief description of some of the Java Standard Libraries:

* **java.lang:** Provides classes that are fundamental to the design of the Java programming language. It is always implicitly being imported as it contains everything as you basically cannot program without String, Double, Enum, Math, etc.

<https://docs.oracle.com/javase/7/docs/api/index.html>

* In **java.util** you can find all the collections and data structures available in Java.

<https://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html>

* Next, we have**java.io** for reading files, working with pipes, streams and similar. It provides for system input and output through data streams, serialization and the file system.

<https://docs.oracle.com/javase/7/docs/api/index.html>

* Also we have **java.nio**, which actually is the alternative to **java.io** and stands for non-blocking I/O. It allows intensive use of the input/output operations as you might guess.

<https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html>

* **java.math** provides functionality for working with arbitrary-precision decimal (BigDecimal) and integer (BigInteger) values.

<https://docs.oracle.com/javase/7/docs/api/java/math/package-summary.html>

* **java.net** is being used for working with sockets, creating connections or in short – creating networking applications

<https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html>

* In Java we also have libraries for working with GUI : **javax.swing** (extension of the older java.awt)

<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

* In **javax.sound.sampled** there are interfaces and classes for capture, processing, and playback of sampled audio data.

<https://docs.oracle.com/javase/7/docs/api/javax/sound/sampled/package-summary.html>

* **java.sql:** Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language.

<https://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html>

* **java.awt.im**: Provides classes and interfaces for the input method

<https://docs.oracle.com/javase/7/docs/api/java/awt/im/package-summary.html>

* **java.awt.font**: Provides classes and interface relating to fonts.

<https://docs.oracle.com/javase/7/docs/api/java/awt/font/package-summary.html>

Apache Commons is actually a whole project focused on creating Java libraries.

Here is a short list of some of the best and most commonly used libraries:

* [**Commons Math**](http://commons.apache.org/proper/commons-math/): The Apache Commons Mathematics Library – the name says it all: this library contains components allowing advanced mathematics and statistics operations and computations

http://commons.apache.org/proper/commons-math/

* [**Commons CLI**](https://commons.apache.org/proper/commons-cli/): provides API for parsing command line arguments. (Why will you even bother creating an application without the ability to pass parameters and control its behavior?)

<https://commons.apache.org/proper/commons-cli/>

* [**Commons CSV**](https://commons.apache.org/proper/commons-csv/): whatever you are developing, at some point you are going to face the necessity of using csv files. This includes opening, reading, editing, saving and creating them. I suggest using the RFC 4180 format from the CSVFormat class and UTF-8 encoding when saving/creating files.

https://commons.apache.org/proper/commons-csv/

* [**Commons IO**](https://commons.apache.org/proper/commons-io/)**:** it is being used for easier execution of input/output operations. Checking at least the ReversedLinesFileReader is definitely worth it.

<https://commons.apache.org/proper/commons-io/>

**Java machine learning libraries**

**MOA** is an open-source software used specifically for machine learning and data mining on data streams in real time. Developed in Java, it can also be easily used with Weka while scaling to more demanding problems. MOA’s collection of machine learning algorithms and tools for evaluation are useful for regression, classification, outlier detection, clustering, recommender systems, and concept drift detection. MOA can be useful for large evolving datasets and data streams as well as the data produced by the devices of the Internet of Things (IoT).

MOA is specifically designed for machine learning on data streams in real time. It aims for time- and memory-efficient processing. MOA provides a benchmark framework for running experiments in the data mining field by providing several useful features including an easily extendable framework for new algorithms, streams, and evaluation methods; storable settings for data streams (real and synthetic) for repeatable experiments; and a set of existing algorithms and measures from the literature for comparison.

<https://moa.cms.waikato.ac.nz/>

**Weka 3** is a fully Java-based workbench best used for machine learning algorithms. Weka is primarily used for data mining, data analysis, and predictive modelling. It’s completely free, portable, and easy to use with its graphical interface. Weka’s collection of machine learning algorithms can be applied directly to a dataset or called from your own Java code. This supports several standard data mining tasks, including data preprocessing, classification, clustering, visualization, regression, and feature selection.

<https://www.cs.waikato.ac.nz/ml/weka/documentation.html>

**Bytecode Libraries**

For writing a framework or libraries that generate code or interact with bytecodes, a bytecode library is needed.

They allow you to read and modify bytecode generated by an application. Some of the popular bytecode libraries in the Java world are javassist and Cglib Nodep.

Javassist (Java Programming Assistant) makes Java bytecode manipulation simple. It is a class library for editing bytecodes in Java; it enables Java programs to define a new class at runtime and to modify a class file when the JVM loads it.

<http://www.javassist.org/>

### ****Messaging Libraries****

Similar to logging and database connection, messaging is also a common feature of many real-world Java applications.

Java provides [JMS](http://javarevisited.blogspot.sg/2014/03/top-10-websphere-mq-series-interview-questions-answers-active-rabbit.html#axzz5Bxv7wony), or the Java Messaging Service, that's not part of JDK. For this component, you need to include a separate  jms.jar.

Similarly, if you are using third-party messaging protocols, like Tibco RV, then, you need to use a third-party JAR —  tibrv.jar — in your application [classpath](http://javarevisited.blogspot.sg/2011/01/how-classpath-work-in-java.html" \t "_blank).

<https://javarevisited.blogspot.com/2014/03/top-10-websphere-mq-series-interview-questions-answers-active-rabbit.html#axzz5Bxv7wony>

### ****PDF Libraries****

Similar to Microsoft Excel, PDF libraries are another ubiquitous format. If you need to support PDF functionality in your application, like [exporting data in PDF files](http://javarevisited.blogspot.sg/2014/05/open-source-java-PDF-File-libraries-Apache-FOP-vs-iText.html), you can use the iText and Apache FOP libraries.

Both provide useful PDF related functionality, but iText is richer and better. See [here](https://www.amazon.com/iText-Action-Covers-5/dp/1935182617?tag=javamysqlanta-20)to learn more about iText.

<https://javarevisited.blogspot.com/2014/05/open-source-java-PDF-File-libraries-Apache-FOP-vs-iText.html>

\*The url given for each library leads to information in greater details about individual library’s documentation which also includes any relevant examples, tutorials and sample codes.

**Submitted by:**

**Samara Chaudhury**

**CSE470 (Section 01)**

**ID: 15305022**