**Glossary of Terms**

**Access Modifier**
Reserved words "public", "private", "protected" in Java. Control whether classes and members may be accessed from any class, only this class, subclasses. Default is access from any class in the package.

**Agile (or Extreme) Development**
Methodology for developing software that emphasizes, among other things, unit testing as part of development process.

**API (Application Programming Interface)**
The way one program uses another program. In Java, the API can be thought of as the collection of public methods for a class or package.

**Class**
Main building block in Java. Contains members, including fields and methods. Classes are the "blueprint" for creating objects.

**Constructor**
Special block of code used to create an instance of a class (or, if you prefer, an object whose type is the class). Used with the "new" keyword (e.g., Person p = new Person() calls the Person() constructor).

**Field**
Member in a class that holds data (e.g., name, age, etc.). Usually marked private so that other programs cannot directly access.

**IDE (Integrated Development Environment)**
Program, like Eclipse, that provides the different tools required to develop a software package.

**JRE (Java Runtime Engine)**
A set of programs that allows you to run Java programs on a specific computer platform, such as Windows, Linux, or Mac OS. The two primary components of the JRE are the Java Virtual Machine (JVM) and the compiled standard Java classes that serve as the API for the Java language.

**JVM (Java Virtual Machine)**
The program that runs Java programs on a specific platform. Java source code is compiled into .class files. These contain the instructions used by the JVM to actually run the programs on a Windows PC, a Linux computer, a Mac computer, etc. The JVM is written for each platform supported by Java. Included in the Java Runtime Engine (or JRE).
JUnit Test
A set of Java classes used to test individual methods in a Java class. Used to build test cases when using agile development methodology.

Method
Member in a class that does some processing (e.g., like a subroutine or function in other languages).

Method Argument, Method Parameter
Parameters refers to the list of variables in a method declaration. Arguments are the actual values that are passed in when the method is invoked. When you invoke a method, the arguments used must match the declaration's parameters in type and order. For example, in the method public setName(String name) {...} "name" is the parameter for this method. If this method is used as follows: myObject.setName("Fred") "Fred" is the argument of the method and it must match the type of the method's parameter.

Object
An instance of a class. For example, Cookie could be a class, and a cookie (e.g. "thisCookie") would be an object created using the class. In other words, "thisCookie" is an object of type Cookie or an instance of Cookie.

Overload (Method)
To provide multiple methods with the same name but different parameters (i.e., same name but different signatures).

Override (Method)
When a subclass implements a method inherited from the super class, this method is said to be overridden.

Package
Packages are imported into a source file to save typing the full name of the class (e.g., can say "Person" instead of "org.eclipse.training.librarytutorial.Person" and to avoid the possibility of two classes having identical names.

Project
In Eclipse, a way to organize your work. An Eclipse workspace can contain multiple projects. Each project can contain multiple packages. Each package can contain multiple classes.

Refactor
To improve a program without changing the way it works (i.e., its API). Example include renaming fields or variables, streamlining code, etc. Very important in agile development because of emphasis on self-documenting code.

Reference Variable
In Java, variable that holds an object reference (e.g., p = new Person();). Points to an area on the "heap" where the object resides. Contrast with value variable.
**Scrapbook Page**
Area in Eclipse where you can execute Java code "snippets" and see how they work. Great for experimenting with Java statements.

**Static Method**
A method that belongs to the entire class instead of one instance of the class. Invoked with `<Class>.<Method>` (e.g., `Person.getTotalCount()`). Used for methods that don't rely on any one instance of a class.

**Swing**
A set of standard Java packages that implement a graphical user interface without using any "native" code.

**SWT (Standard Widget Toolkit)**
Set of Java classes and native programs developed by Eclipse to allow Java programs to have the look and feel of native programs on each platform. Used to create the Eclipse IDE.

**Type**
In Java, an attribute of a variable to indicate either a primitive type (int, boolean, etc.) or class membership. For objects, the type is the class to which it belongs. Types also include interfaces and enumerations.

**Value Variable**
In Java, variable that holds the value of a Java primitive (e.g., integer, character, etc.). Held in the memory stack. Contrast with reference variable.

**Workspace**
Top-level container for Eclipse work. Holds multiple projects. In a single Eclipse session, only one workspace can be active.