**Text 1 (Lecture slides)**

A nested loop can be used to sort an array into a specific order. For example, an array of integers can be sorted into an ascending or descending order. Consider the program on the next slide which implements a basic (inefficient) sort called bubblesort which sorts an array into ascending order. This algorithm is called bubbles or because in order to sort the array it ‘bubbles’ values ‘up’ the array, i.e. moves them along until they are in the correct position.

Source © Ian Kenny

**Example 2 (Online Java tutorial)**

A constructor initializes an object when it is created. It has the same name as its class and is syntactically similar to a method. However, constructors have no explicit return type. Typically, you will use a constructor to give initial values to the instance variables defined by the class, or to perform any other start-up procedures required to create a fully formed object. All classes have constructors, whether you define one or not, because Java automatically provides a default constructor that initializes all member variables to zero. However, once you define your own constructor, the default constructor is no longer used.

Source: <https://www.tutorialspoint.com/java/java_constructors.htm>

**Example 3 (Academic paper)**

Java modules assemble packages, classes, native code, and further resources, like simple JAR files. Yet, the new modules contain a static module descriptor which specifies the module’s unique name, its dependency on other modules, its exported packages, and a definition of re-exported dependencies. The module descriptor is processed by the Java compiler as well as the Java Virtual Machine (JVM), causing them to check and prevent access to the internal types of a module both at compile- and run-time. The dependencies between modules, as specified in the module descriptors, form an acyclic module graph. This module graph is used to resolve references between classes, replacing the previous class-loading based on the linear classpath.

Source: A. Dann, B. Hermann and E. Bodden, (2019). ModGuard: Identifying Integrity & Confidentiality Violations in Java Modules, IEEE Transactions on Software Engineering, 42, in press.