CMSC131

Java Memory Model Recap

(posted for reading as we delve deeper into objects and statics)

The Java Memory Model

There are a ***wide variety*** of things to explore to learn about the entire Java Memory Model.

In this course we have been and will continue to focus on:

- primitive variables, references to objects, and objects
- **stack** versus **heap** versus **metaspace**
- single-threaded programs

Stack - Heap - Metaspace

The **stack** and the **heap** and **metaspace** are three logical areas within a computer's memory where information is stored by a Java Virtual Machine.

They each have a different purpose in a typical Java program.

Stack

The **stack** can be seen as the part of memory where method parameters and local variables (both primitives and references) are stored when a method is invoked and while it runs.

As each method is invoked a new **stack frame** is created. As each method ends, its frame is "popped" off the stack. The information there can no longer be accessed.

Heap

The **heap** can be thought of as the part of the JVM's memory where objects live.

Unlike the **stack**, things added to the **heap** don't have to be removed in reverse order and they can persist and be accessed beyond the end of a method call as long as references to them are still available.

Garbage Collection

The short version is that the areas in memory where objects are stored on the **heap** that are no longer referenced by anything that can be reached are available to be released for reuse by new objects as they are created.

Some languages require the programmer to explicitly mark objects for deletion. Java chooses to manage deletion on its own to reduce memory leaks.

Metaspace (PermGen)

In an update to the Java Memory Model in 2004, the "Permanent Generation" area of memory was added to the two existing areas (**stack** and **heap**).

It was decided that things that never went away, such as static fields, would be stored there. These static fields would have space allocated for them as soon as the JVM knew that the program would use their class.

One of the changes in Java 8 was that some of the implementation details and uses of this space were modified and it was renamed **metaspace**.