Runtime Data Areas

**What Are Runtime Data Areas?** Here is the definition the JVM specification: "The Java Virtual Machine defines various run-time data areas that are used during execution of a program. Some of these data areas are created on Java Virtual Machine start-up and are destroyed only when the Java Virtual Machine exits. Other data areas are per thread. Per-thread data areas are created when a thread is created and destroyed when the thread exits."

There are 6 JVM data areas divided into 2 categories:

* Thread shared data areas: Method Area, Heap and Direct Memory - These data areas are created at JVM start-up time and shared by all threads.
* Per-thread data areas: PC Register, JVM Stack, Native Method Stack - These data areas are created when a thread is created and used by that thread only. If multiple threads are running, multiple copies of these data areas will be created.

Here a shot description of each JVM runtime data area:

1. Method Area - A thread shared data area that stores per-class structures such as the run-time constant pool, field and method data, and the code for methods and constructors.

2. Heap - A thread shared data area that stores all data objects including both class instances and arrays, except those allocated explicitly in Method Area and Direct Memory area. Objects in heap are allocated when explicitly requested by the application and deallocated by JVM garbage collector automatically.

3. Direct Memory - A thread shared data area that stores buffer objects allocated explicitly in direct memory area. Objects in direct memory are also automatically deallocated by JVM garbage collector.

4. PC (Program Counter) Register - A per-thread data area that contains the address of the instruction currently being executed in the thread. If the method currently being executed is native, the value of the PC register is undefined.

5. JVM Stack - A per-thread data area that stores a stack of frames. Each frame holds local variables and partial results, and plays a part in method invocation and return of the current running method.

6. Native Method Stack - A per-thread data area that stores similar data elements as a JVM stack to execute native (non-Java) methods.