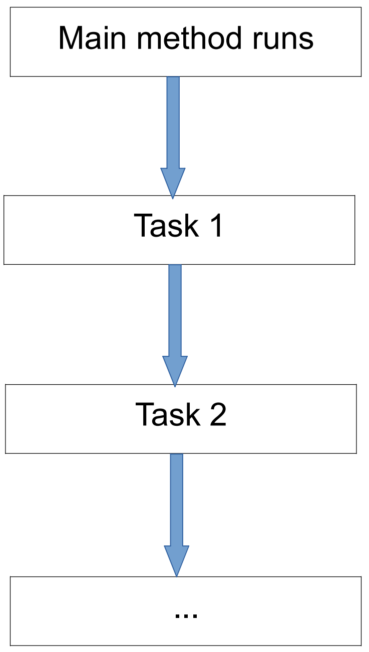
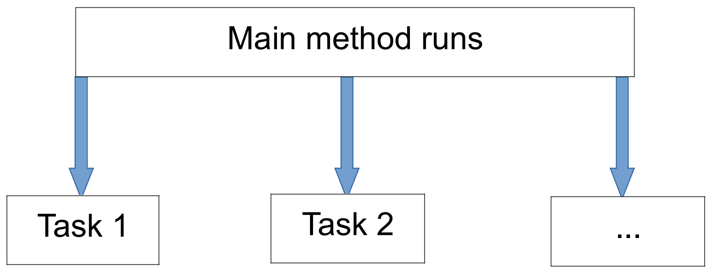
A simple program



The main method will run and then wait for every process, for simple programs this might not be an issue but the more complex and the more resource expensive the process the more it will effect the usability of the program. An example of this is the reading and creation of the menu, price and stock collections in the cash register exercise (/javabasics/cashregister/solutions/task1/Main.java). The BufferedReaders are expensive and will block the main thread until the tasks are complete.

This is a good place to consider Multithreading. Java provides the feature of Multithreading that allows concurrent execution of multiple processes to maximize the use of your CPU by the program. Multithreading can improve your program but it comes with overhead that has to be considered by the developer. All threads running in a program share the same memory and this can cause concurrency issues. If one thread writes values to a shared resource and another tries to read from it at the same time could produce unexpected results. If you choose to add Multithreading to your program always be sure to consider concurrency.

A multithreaded program



<u>Resources:</u>

multithreading theory and tutorials:
https://www.javatpoint.com/multithreading-in-java

ExecutorService is a framework provided by the JDK which simplifies the execution of tasks asynchronously
https://www.baeldung.com/java-executor-service-tutorial

runnable vs callable Both interfaces are designed to represent a task that can be executed by multiple threads:
https://www.baeldung.com/java-runnable-callable