I spent my first day on the job searching for a main method. I had found a few, but they didn't seem to do anything. Some module objects were being created and that's it.

How was the rest of the code running? Who was creating all the handler objects? Where was the while(true) to keep the program running forever?

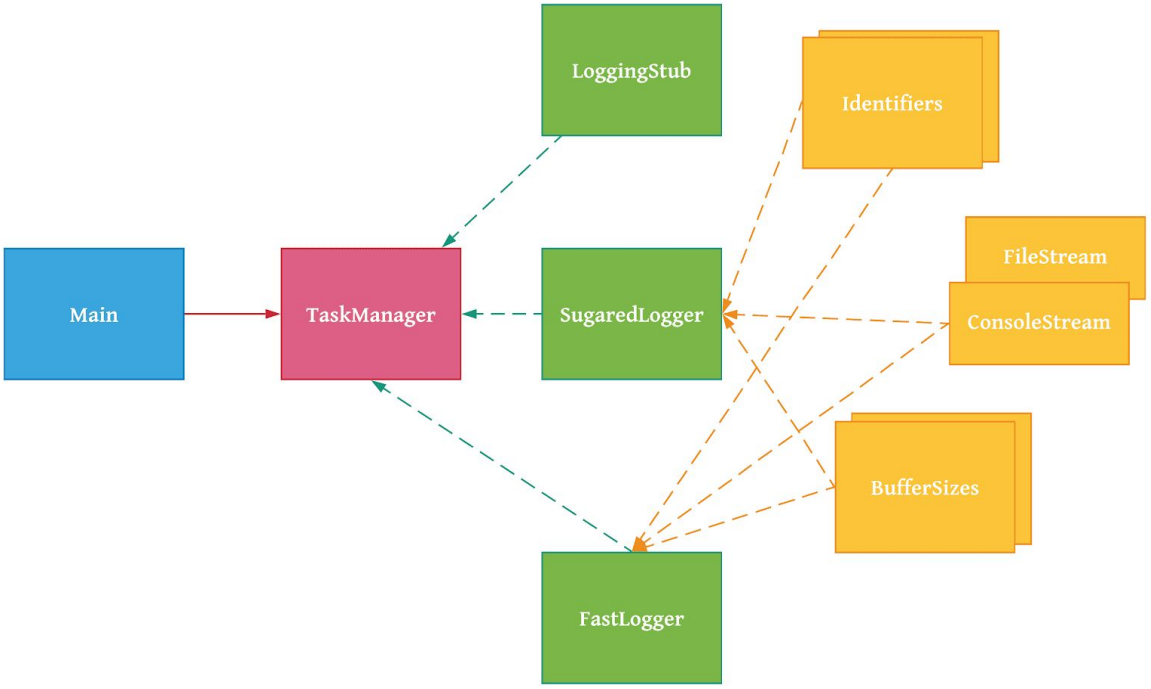
For example, an object create with the new keyword may look like:

TaskManager taskManager = ***new TaskManager(new SugaredLogger(System.out, "sugared-service: "));***

This introduces tight coupling between the TaskManager's construction and the SuggaredLogger. At the site of creating this object, *we have to be aware of which OutputStream and what identifier to use for a dependency*.

Compare this with:
TaskManager taskManager = ***injector.getInstance(TaskManager.class)***

This is called dependency injection. The class creating TaskManager does not need to know how it is created. It just asks for one, and gets it.



The configuration and dependencies are handled by a Factory.
DI is an elaborate use of a software pattern called the Factory Pattern.

1) Allowing multiple configurations (By separating configuration from usage)

2) Complex instantiations are done by the factory.
3) Avoids hard-coding in the code. That avoids redeployments for property changes.
4) Writing tests with stubbing is easier.

Disadvantages:

1) Confusing to a new engineer (Code flow is difficult to visualise).
2) It isn't obvious what properties are set in code.