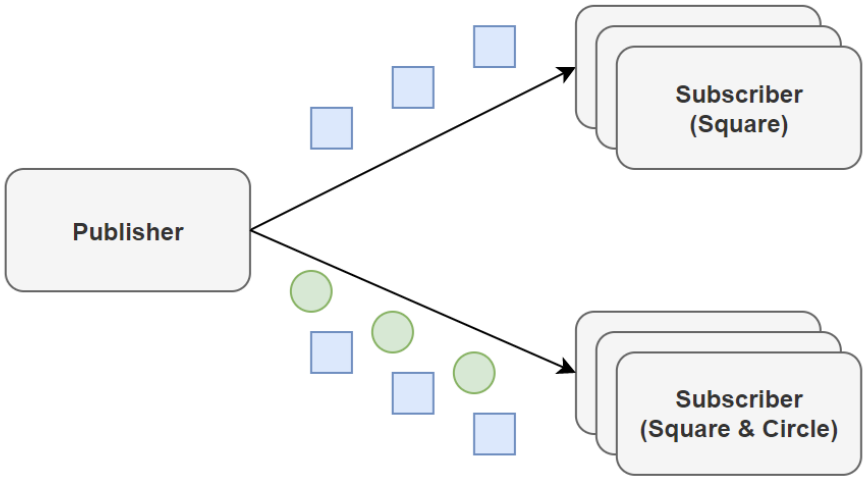
omnicon

Data Distribution Exercise

User Story:
The exercise intended to build a C++/Python-based data distribution system with various connectivity/application-level functionalities.

Top-level architecture & Requirements:

- A single publisher (pub) application publishes data on blue square (at 1 Hz) and green circle (at 2Hz).
- Two kinds of the subscriber (sub) applications subscribe/receive the data:
    - One kind is only interested in squares.
    - One kind is interested in squares and circles.
- It is possible to initialize numerous subscribers from each kind.



- The communication between the pub and subs is UDP based.
    - All applications are required to run on the same machine/PC for now.
- Each shape (i.e. square and circle) has the following properties:
    - Color – can be blue or green (according to the illustration below).
    - Size – fixed size.
    - Lat, long, alt coordinates – can be filled for random values.
- The subscriber is required to print each message upon reception (message format/serialization is to the developer decision)

General Guidelines:

- Use standardized code styling OO best practices:
    - Classes and Objects
    - Modules and Packages
    - Exception handling
    - Type hints and function return values
- Utilize modern C++/Python functionality as much as possible.
- Utilize C++/Python features as much as possible (Data abstraction, Data encapsulation, Inheritance, Data hiding, Polymorphism).
- Avoid using OS-specific APIs.
- Document your code.
- Keep the code simple (do not over complicate).
- Provide a basic class diagram of the solution (you can use Drawio for this).