[Problem Solving] Driver Allocation OOD

Rules of the Game

1. You are free to choose your best language and IDE (Development Environment) as per your choice.
2. We are particularly interested in understanding your craft and how you design your solution. Your code enables us to know about your ability to code. So the key things we are interested are:
    - The final code should compile and run successfully.
    - All corner cases should be handled.
    - The code is modular with clear interfaces identified and coded.
    - Proper use of abstractions and clean production ready code.
    - Code can be extended for features addition.

Problem Definition

Gojek started as a call center that receives bookings from calls. Once the executive receives a call, he will need to dispatch a driver to the booking. If all drivers are occupied, the executive needs to tell the customer. A new driver needs to be registered to the system before he is available on the system. I want to create an automated booking management system that allows the executive to dispatch drivers without human intervention with the **following features**.

1. When a booking request comes in, I want to have a driver assigned to the booking given a drive is available and return a booking_id in the response.
2. Distance information is needed on the booking creation.
3. A driver can only take a single booking at a time.
4. Once the driver completes the booking, they will notify the executive to make the driver available in the pool.
5. The driver assignment is randomized.

Scope

We interact with the system via a simple set of commands which produce a specific output. Please take a look at the example below, which includes all the commands you need to support - they're self-explanatory. The system should allow input in two ways. Just to clarify, the same codebase should support both modes of input - we don't want two distinct codebases.

1. It should provide us with an interactive command prompt based shell where commands can be typed in. And the supported commands are

```
$ register_driver <<driver_id>>
$ dispatch_driver_for_a_booking <<booking distance>>
$ complete_booking <<booking-id>>
```

The sample working cli will look like:

```
$ register_driver driver-001
Driver driver-001 registered

$ register_driver driver-002
Driver driver-002 registered

$ dispatch_driver_for_a_booking 4
Driver driver-001 is assigned to booking booking-1 with 4 km distance

$ dispatch_driver_for_a_booking 15
Driver driver-002 is assigned to booking booking-2 with 15 km
distance

$ dispatch_driver_for_a_booking 1
Sorry, driver is not available

$ complete_booking booking-1
Driver driver-001 is released to allocation pool

$ dispatch_driver_for_a_booking 1
Driver driver-001 is assigned to booking booking-3 with 1 km distance
```

```
$ complete_booking booking-3
Driver driver-001 is released to allocation pool

$ dispatch_driver_for_a_booking 1
Driver driver-001 is assigned to booking booking-4 with 1 km distance

$ complete_booking booking-4
Driver driver-001 is released to allocation pool
```

We are particularly interested in understanding your craft and how you design your solution. Your code enables us to know about your ability to code. So the key things we are interested are:

- The final code should compile and run successfully.
- All corner cases should be handled.
- The code is modular with clear interfaces identified and coded.
- Proper use of abstractions and clean production ready code.
- Code can be extended for features addition.

Valid Assumptions

As a part of the scope of the above-defined problem statement you can proceed with the following assumptions:

1. **You don't need to over-optimize the time complexity of the code.**

Problem Extension

There are 2 parts of the extension:

1. As appreciation of the driver's hard work, I want to give some incentive when they achieve certain conditions. I want the system to provide the ability to find out drivers with:
   - Booking completed >= 3
   - Cumulative booking distance > 10 km

```
$ drivers_completed_booking_gt 3
Driver Completed
driver-001 3

$ booking_completed_distance_gt_10
Driver Completed

$ complete_booking booking-2
Driver driver-002 is released to allocation pool

$ booking_completed_distance_gt_10
Driver Completed
driver-002 10km
```

2. It should accept a filename as a parameter at the command prompt and read the commands from that file