**Batch: A3 Roll No.: 16010123085**

**Experiment / assignment / tutorial No.\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Implement Object-Oriented Programming in Dart. |

**AIM:** To implement the concepts of **Object-Oriented Programming (OOP)** in Dart such as **classes, objects, constructors, inheritance, polymorphism, encapsulation, and abstraction**.

**Objective:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1 : Understand basics, object-oriented programming concepts and Null safety in Dart & Flutter.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O’ Reilly 2 nd Edition, 2016.

**Problem Statement:**

**A) Bank Account Management System**

**Objective:**

Implement a Dart program that uses **encapsulation** to protect sensitive data and **inheritance** to create specific types of bank accounts.

**Requirements:**

**1. Encapsulation:**

Create a base class BankAccount with the following **private data members**:

* \_accountNumber (String)
* \_accountHolderName (String)
* \_balance (double)

Provide **getters** for account number and holder name, and both **getter and setter** for balance (but restrict direct negative balance using setter logic).

**2. Inheritance:**

Create two subclasses:

* SavingsAccount
  + Add interest to balance (method: addInterest(double rate))
* CurrentAccount
  + Deduct monthly maintenance charge (method: deductMonthlyCharge(double charge))

**3. Demonstration:**

* Create objects of SavingsAccount and CurrentAccount.
* Perform deposit, withdraw, and subclass-specific operations.
* Print updated details using a method displayAccountInfo().

**B) Event Management Using Dart and OOP Concepts**

Objective:

Design and implement a console-based application in Dart to simulate an event management system. Students must apply core object-oriented programming principles like abstraction, inheritance, polymorphism, and encapsulation.

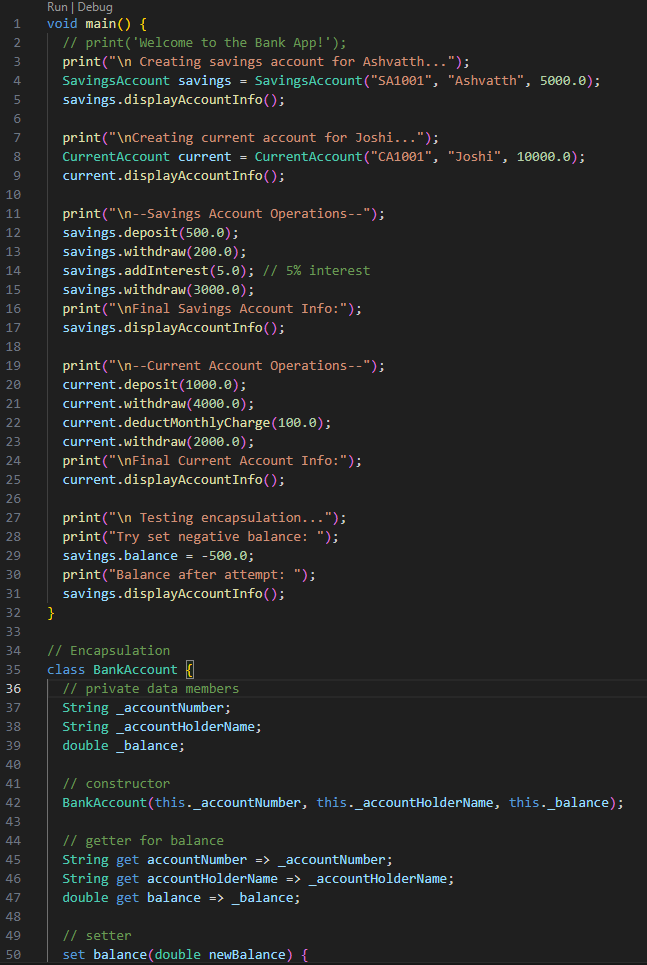
Develop a command-line Dart application to manage events such as **conferences**, **workshops**, and **meetups**. The application should allow administrators to:

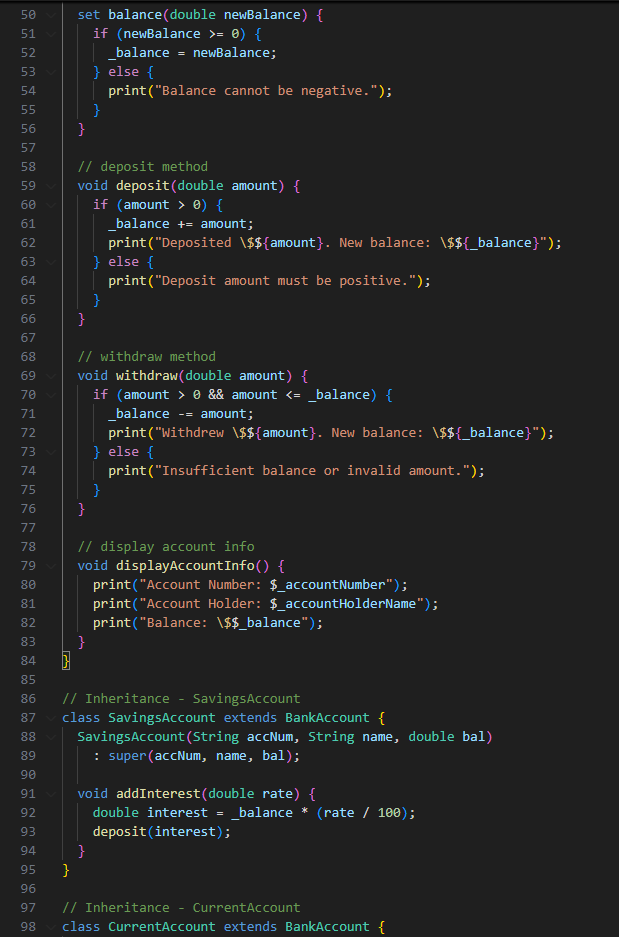
* Add new events
* Register participants
* Display event and participant information
* Filter and sort events using functional programming tools
* Calculate statistics (e.g., total attendees per event type)

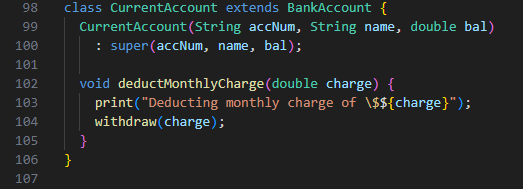
**Implementation Details:**

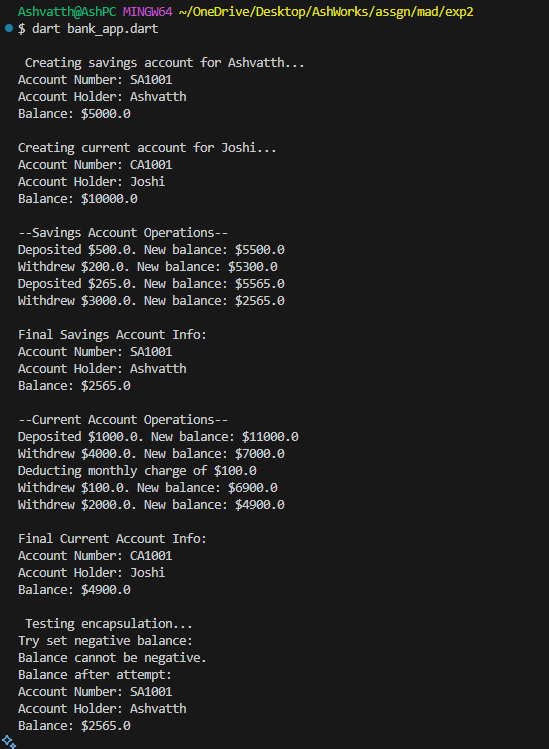
A. Bank Account Management System

Code:



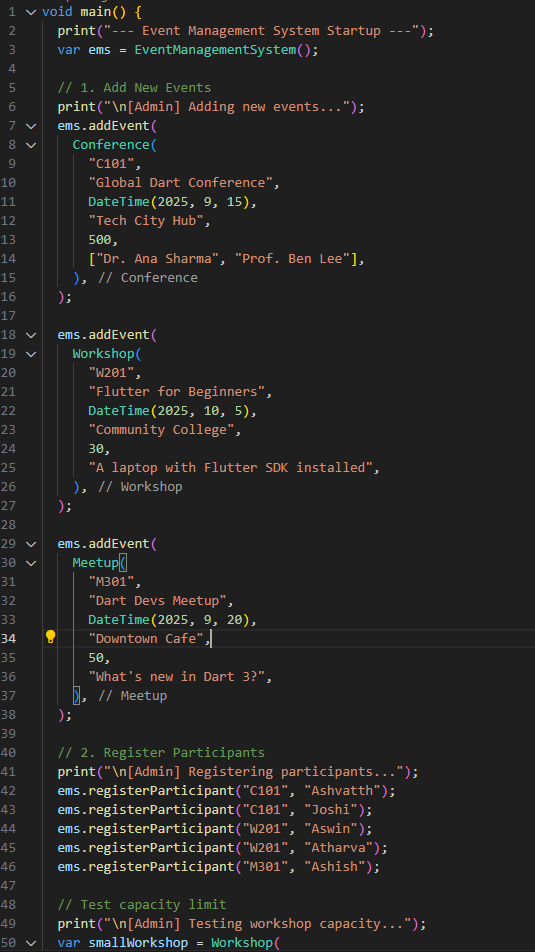


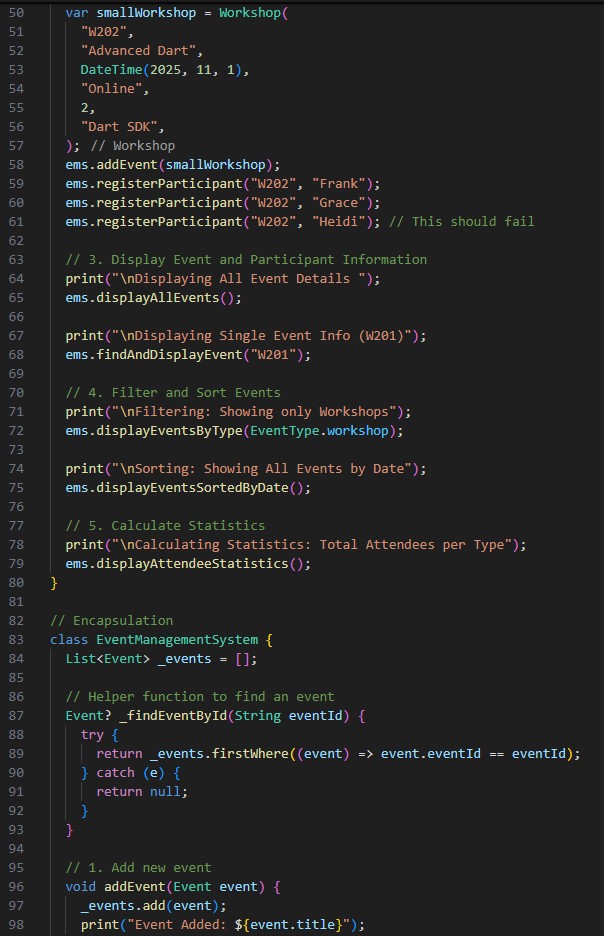
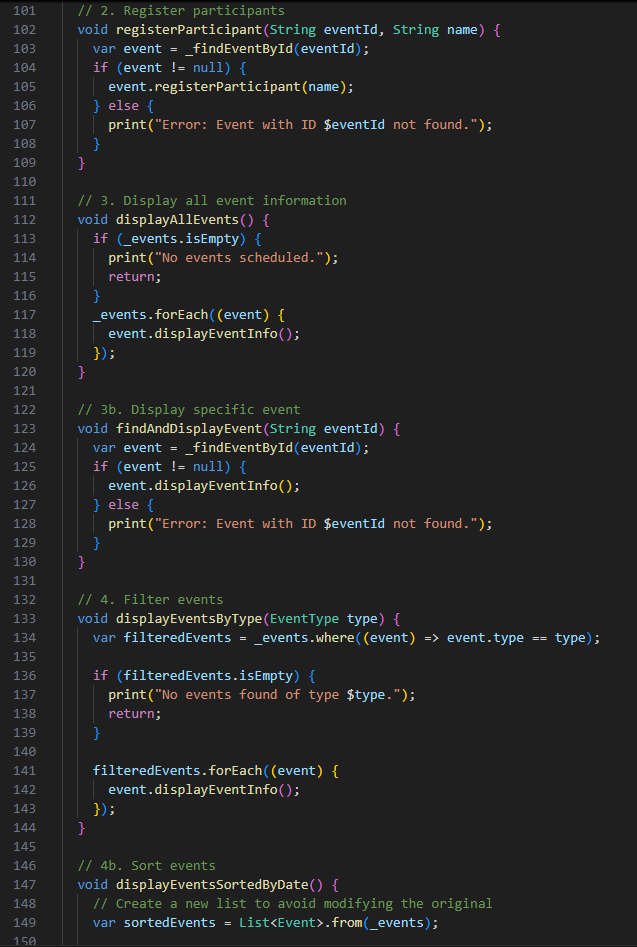
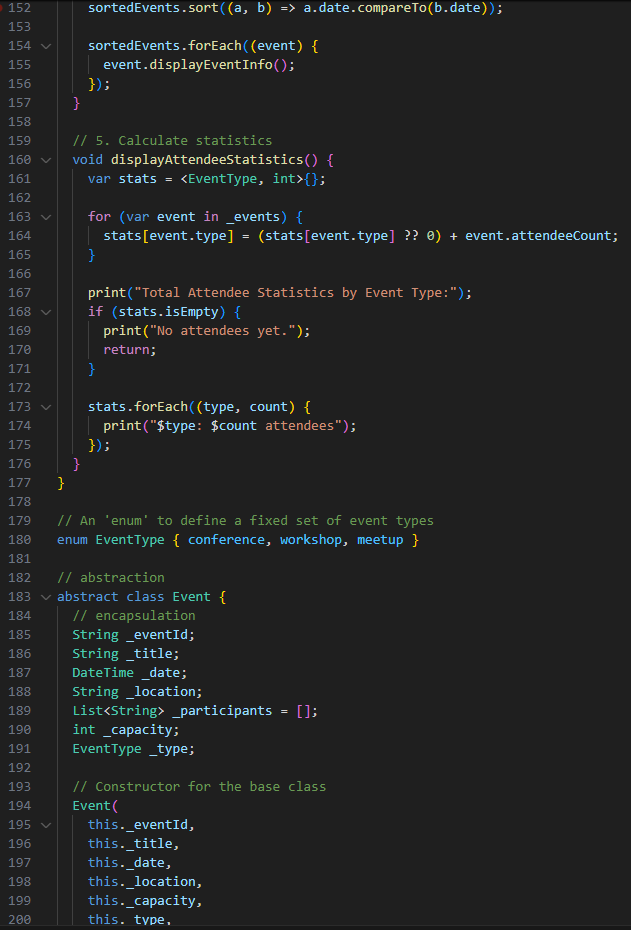
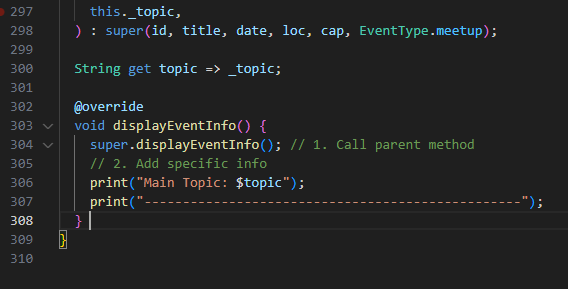


Output:  


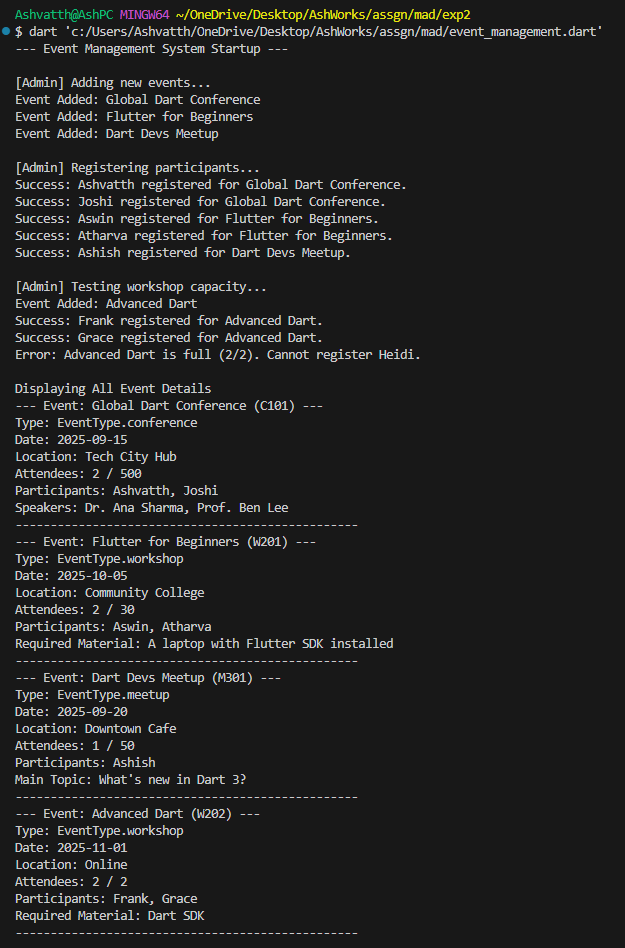
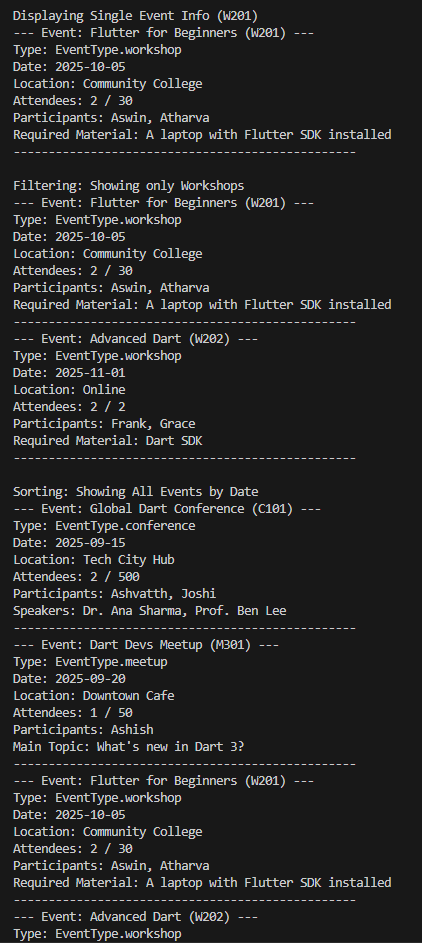
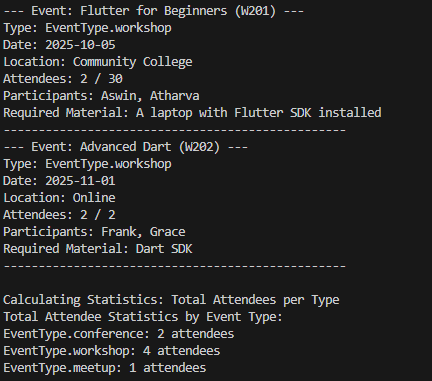
B. Event Management:

Code:



Output:

**Conclusion:**

Used OOP concepts in dart language through 2 exercises:

* 1. Showed how encapsulation protects data and inheritance creates specialized reusable classes
  2. Expanded on this using abstraction to define a template, inheritance for varied event types, and polymorphism to manage all events in a single list, proving OOP’s power in building complex systems.