AIM:

Lab-2:
Dart Programming Language
Learning Resources - Introductory Dart Video (Last 2 hrs.)

Practical Task – Library Management System
Objective:
Develop a simple console-based library management system for a book collection. The system should allow users to perform various operations while demonstrating the use of conditional statements, loops, functions, classes, inheritance, exception handling, and async programming.Task Requirements:
Book Class:

- Create a Book class with properties: title, author, yearPublished, and isAvailable (boolean).
- Implement getters and setters for these properties.
- Include a method to display book details.

Library Class:

- Create a Library class that maintains a collection of Book objects.
- Include a method to add a book to the library.
- Include a method to borrow a book (mark it as not available).
- Include a method to return a book (mark it as available).
- Use a constructor to initialize the library with a predefined list of books.

User Interaction:

- Use a loop to present a menu to the user with the following options:
- Add a new book.
- Borrow a book.
- Return a book.
- List all books.
- Exit the system.

Use conditional statements to handle user inputs and navigate the menu.Switch-Case for Menu Operations:

- Use a switch-case statement to process menu selections.

Handling Exceptions:

- Implement exception handling for cases like attempting to borrow a book that is unavailable or returning a book that was not borrowed.

Inheritance:

- Create a subclass EBook that inherits from Book and adds a property fileSize. Override the method that displays book details to include the file size.

Static Methods:

- Implement a static method in the Library class that keeps track of the total number of books in the library.

Abstract Class:

- Create an abstract class User with an abstract method displayUserType(). Create a subclass Member that implements this method.

Use of Async and Await:

- Simulate an asynchronous process for listing books with a method in the Library class that returns a Future<List<Book>> and uses await to simulate a delay.

THEORY:

Comprehensive Overview of Dart Syntax
Dart is a modern, object-oriented programming language optimized for building fast applications. Below is an overview of its syntax, structured into categories.

1. Basic Syntax
Variables

- Dart supports var, final, const, and explicit types.

```
var name = "John";          // Type inferred
String city = "New York";   // Explicit type
final age = 25;             // Immutable, runtime constant
const pi = 3.14159;         // Compile-time constant
```

Data Types

- Dart includes basic types: int, double, String, bool, List, Set, Map.

```
int a = 10;
double b = 3.14;
String greeting = "Hello";
bool isLoggedIn = true;
```

Printing Output
dart
Copy code

```
print("Hello, Dart!");
```

2. Control Flow Statements
Conditional Statements

- if, else, and else if blocks for decision-making.

```
if (a > b) {
  print("a is greater");
} else if (a == b) {
  print("a and b are equal");
} else {
  print("b is greater");
}
```

Switch-Case

- Ideal for handling multiple conditions.

```
switch (day) {
  case 'Monday':
    print('Start of the week');
    break;
  case 'Friday':
    print('End of the workweek');
    break;
  default:
    print('Another day');
}
```

Loops

- **For Loop:**

```
for (int i = 0; i < 5; i++) {
  print(i);
}
```

- **While Loop:**

```
int i = 0;
while (i < 5) {
  print(i++);
}
```

- **Do-While Loop:**

```
int i = 0;
```

```
do {
  print(i++);
} while (i < 5);
```

- **For-In Loop:**

```
var numbers = [1, 2, 3];
for (var num in numbers) {
  print(num);
}
```

3. Functions

Basic Function

```
int add(int a, int b) {
  return a + b;
}
```

Arrow Functions

- A concise syntax for one-line functions.

```
int square(int x) => x * x;
```

Optional Parameters

- **Positional Parameters:**

```
void greet(String name, [String title = "Mr/Ms"]) {
  print("Hello, $title $name!");
}
```

- **Named Parameters:**

```
void display({String? title, int? year}) {
  print("$title was released in $year.");
}
}
```

4. Object-Oriented Programming

Classes and Objects

```
class Person {
  String name;
  int age;

  Person(this.name, this.age);

  void display() {
    print("Name: $name, Age: $age");
  }
}

var person = Person("Alice", 25);
person.display();
```

Inheritance

```
class Employee extends Person {
  String jobTitle;

  Employee(String name, int age, this.jobTitle) : super(name, age);
}
```

Abstract Classes

```
abstract class Shape {
  void draw();
}

class Circle extends Shape {
  void draw() {
    print("Drawing a circle");
  }
}
```

5. Collections

List

```
List<int> numbers = [1, 2, 3];
numbers.add(4);
print(numbers);
```

Set

```
Set<String> fruits = {"Apple", "Banana", "Cherry"};
fruits.add("Apple");  // Duplicates are ignored
print(fruits);
```

Map

```
Map<String, int> ageMap = {"Alice": 30, "Bob": 25};
print(ageMap["Alice"]);
```

6. Exception Handling

```
try {
  int result = 10 ~/ 0; // Throws an exception
} on IntegerDivisionByZeroException {
  print("Cannot divide by zero");
} catch (e) {
  print("Error: $e");
} finally {
  print("Cleanup actions");
}
```

7. Async and Await

Future

```
Future<String> fetchData() async {
  return "Data loaded";
}

fetchData().then((data) => print(data));
```

Async-Await

```
Future<void> loadData() async {
  var data = await fetchData();
  print(data);
}
```

8. Miscellaneous Features

Null Safety

```
String? nullableVar; // Can be null
nullableVar = "Not Null";
```

Type Casting

```
dynamic value = "Hello";
String str = value as String;
```

Static Members

```
class Example {
  static int count = 0;
  static void displayCount() {
    print("Count: $count");
  }
}
Example.displayCount();
```

Getters and Setters

```
class Car {
  String _model = "";

  String get model => _model;
  set model(String model) => _model = model; }
```

CODE:

```
import 'dart:async';
import 'dart:io';

class Book {
  int _bookId;
  String _title;
  String _author;
  int _yearPublished;
  bool _isAvailable;

  Book(this._bookId, this._title, this._author, this._yearPublished, this._isAvailable);

  // Getters and Setters
  int get bookId => _bookId;
  set bookId(int id) => _bookId = id;

  String get title => _title;
  set title(String newTitle) => _title = newTitle;

  String get author => _author;
  set author(String newAuthor) => _author = newAuthor;

  int get yearPublished => _yearPublished;
  set yearPublished(int year) => _yearPublished = year;

  bool get isAvailable => _isAvailable;
  set isAvailable(bool availability) => _isAvailable = availability;

  // Method to display details
  void displayDetails() {
    print('Book ID: $_bookId');
    print('Title: $_title');
    print('Author: $_author');
    print('Year Published: $_yearPublished');
    print('Available: ${_isAvailable ? "Yes" : "No"}\n');
  }
}

class EBook extends Book {
  double _fileSize;

  EBook(int bookId, String title, String author, int yearPublished, bool isAvailable,
this._fileSize)
      : super(bookId, title, author, yearPublished, isAvailable);

  double get fileSize => _fileSize;
  set fileSize(double size) => _fileSize = size;

  @override
```

```
  void displayDetails() {
    super.displayDetails();
    print('File Size: ${_fileSize}MB\n');
  }
}

abstract class User {
  void displayUserType();
}

class Member extends User {
  @override
  void displayUserType() {
    print("User Type: Member");
  }
}

class Library {
  final List<Book> _books = [];
  static int _totalBooks = 0;

  Library() {
    // Predefined books
    _books.add(Book(1, "Dart Basics", "John Doe", 2020, true));
    _books.add(Book(2, "Flutter Advanced", "Jane Smith", 2021, true));
    _books.add(EBook(3, "Async Programming", "Alex Brown", 2022, true, 1.5));
    _totalBooks = _books.length;
  }

  // Getters
  List<Book> get books => _books;

  static int get totalBooks => _totalBooks;

  // Methods
  void addBook(Book book) {
    _books.add(book);
    _totalBooks++;
    print("Book added successfully!\n");
  }

  void borrowBook(int bookId) {
    try {
      var book = _books.firstWhere((b) => b.bookId == bookId);
      if (book.isAvailable) {
        book.isAvailable = false;
        print("You borrowed '${book.title}'\n");
      } else {
        throw Exception("Book is currently unavailable!");
      }
    } catch (e) {
```

```
      print("Error: $e\n");
    }
  }

  void returnBook(int bookId) {
    try {
      var book = _books.firstWhere((b) => b.bookId == bookId);
      if (!book.isAvailable) {
        book.isAvailable = true;
        print("You returned '${book.title}'\n");
      } else {
        throw Exception("Book was not borrowed!");
      }
    } catch (e) {
      print("Error: $e\n");
    }
  }

  Future<void> listBooks() async {
    print("Fetching book list...");
    await Future.delayed(Duration(seconds: 2));
    print("\n--- Book List ---");
    for (var book in _books) {
      book.displayDetails();
    }
  }

  static void displayTotalBooks() {
    print("Total books in the library: $_totalBooks\n");
  }
}

void main() async {
  Library library = Library();

  while (true) {
    print("=== Library Menu ===");
    print("1. Add a Book");
    print("2. Borrow a Book");
    print("3. Return a Book");
    print("4. List All Books");
    print("5. Display Total Books");
    print("6. Exit");
    print("====================");
    stdout.write("Enter your choice: ");
    int choice = int.parse(stdin.readLineSync()!);

    switch (choice) {
      case 1:
        stdout.write("Enter Book ID: ");
        int id = int.parse(stdin.readLineSync()!);
```

```
      stdout.write("Enter Title: ");
      String title = stdin.readLineSync()!;
      stdout.write("Enter Author: ");
      String author = stdin.readLineSync()!;
      stdout.write("Enter Year Published: ");
      int year = int.parse(stdin.readLineSync()!);
      stdout.write("Enter File Size (Enter 0 for physical books): ");
      double fileSize = double.parse(stdin.readLineSync()!);

      if (fileSize > 0) {
        library.addBook(EBook(id, title, author, year, true, fileSize));
      } else {
        library.addBook(Book(id, title, author, year, true));
      }
      break;
    case 2:
      stdout.write("Enter Book ID to borrow: ");
      int bookId = int.parse(stdin.readLineSync()!);
      library.borrowBook(bookId);
      break;
    case 3:
      stdout.write("Enter Book ID to return: ");
      int bookId = int.parse(stdin.readLineSync()!);
      library.returnBook(bookId);
      break;
    case 4:
      await library.listBooks();
      break;
    case 5:
      Library.displayTotalBooks();
      break;
    case 6:
      print("Exiting system. Goodbye!");
      return;
    default:
      print("Invalid choice! Please try again.\n");
  }
 }
}
```

OUTPUT:

```
#3      _RawReceivePort._handleMessage (dart:isolate-patc

C:\Users\SMIT PATEL\Desktop\mad\myapp>dart run
Building package executable...
Built myapp:myapp.
=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 123
Invalid choice! Please try again.

=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 1
Enter Book ID: 123
Enter Title: sddf
Enter Author: aqsdf
Enter Year Published: 1234
Enter File Size (Enter 0 for physical books): 12
Book added successfully!
```

```
39    class EBook extends Book {
PROBLEMS 7   OUTPUT   TERMINAL   PORTS   COMMENTS

6. Exit
====================
Enter your choice: 1
Enter Book ID: 123
Enter Title: sddf
Enter Author: aqsdf
Enter Year Published: 1234
Enter File Size (Enter 0 for physical books): 12
Book added successfully!

=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 2
Enter Book ID to borrow: 123
You borrowed 'sddf'

=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 3
```

```
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 4
Fetching book list...

--- Book List ---
Book ID: 1
Title: Dart Basics
Author: John Doe
Year Published: 2020
Available: Yes

Book ID: 2
Title: Flutter Advanced
Author: Jane Smith
Year Published: 2021
Available: Yes

Book ID: 3
Title: Async Programming
Author: Alex Brown
Year Published: 2022
Available: Yes

File Size: 1.5MB
```

```
File Size: 1.5MB

Book ID: 123
Title: sddf
Author: aqsdf
Year Published: 1234
Available: Yes

File Size: 12.0MB

=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice: 5
Total books in the library: 4

=== Library Menu ===
1. Add a Book
2. Borrow a Book
3. Return a Book
4. List All Books
5. Display Total Books
6. Exit
====================
Enter your choice:
```

Latest Applications:

1. **Library Management Systems (LMS):**
   - Forms the basis for more complex LMS applications used in schools, colleges, and public libraries.
2. **E-Book Platforms:**
   - The EBook class showcases how digital resources can be managed alongside

physical books, a key feature in platforms like Kindle or OverDrive.

3. **Inventory Systems:**
   - The logic can be adapted to manage inventory in retail, warehouses, or online stores.
4. **Learning Management Systems:**
   - With modifications, this system could manage educational content, such as course materials and user access, in online learning platforms like Moodle or Blackboard.
5. **Async Operations for Cloud Apps:**
   - The asynchronous functionality mimics operations in cloud-based systems, such as fetching large data from a database.
6. **Mobile App Backends:**
   - Acts as a prototype for backend services managing books or digital content for mobile apps developed using Flutter or Dart.
7. **Collaborative Libraries:**
   - Could be extended to support shared libraries for communities, promoting resource sharing in collaborative ecosystems.
8. **Integration with Modern Tech**:
   - Can integrate with databases (e.g., Firebase or MongoDB) and APIs for a fully functional web or mobile application.

Learning Outcome:

1. **Object-Oriented Programming (OOP):**
   - Understand the principles of OOP, including encapsulation, inheritance, abstraction, and polymorphism.
   - Learn to design reusable and modular code using classes and objects.
2. **Encapsulation:**
   - Implement private properties and access them using getters and setters to maintain data integrity.
3. **Inheritance and Polymorphism:**
   - Explore inheritance by creating a subclass (EBook) and overriding methods to extend functionality.
4. **Exception Handling:**
   - Handle runtime errors gracefully, such as borrowing an unavailable book or returning a non-borrowed book.
5. **Asynchronous Programming:**
   - Use Future and await for simulating asynchronous operations, improving program responsiveness.
6. **Static Methods and Properties:**
   - Understand the concept of class-level methods and variables, such as tracking total books.
7. **Abstract Classes:**
   - Learn abstraction by creating an abstract User class and implementing it in the Member subclass.
8. **Switch-Case for Flow Control:**
   - Use switch-case for efficient menu navigation and user interaction.
9. **Real-World Problem Solving:**
   - Apply theoretical programming concepts to develop a practical system that mimics real-world applications.