**Flutter Assignment**

**Module – 2**

**Flutter – Dart Programing**

**//11. Write a Program to check the given year is leap year or not.**

import 'dart:io';

void main() {

print("Enter a year:");

int year = int.parse(stdin.readLineSync()!);

// Check if the year is a leap year

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

print("$year is a leap year.");

} else {

print("$year is not a leap year.");

}

}

**//12. Write a Program to check the given number is prime or not prime.**

import 'dart:io';

void main() {

print("Enter a number:");

int number = int.parse(stdin.readLineSync()!);

bool isPrime = checkPrime(number);

if (isPrime) {

print("$number is a prime number.");

} else {

print("$number is not a prime number.");

}

}

bool checkPrime(int n) {

if (n <= 1) {

return false;

}

if (n == 2 || n == 3) {

return true;

}

if (n % 2 == 0 || n % 3 == 0) {

return false;

}

int i = 5;

while (i \* i <= n) {

if (n % i == 0 || n % (i + 2) == 0) {

return false;

}

i += 6;

}

return true;

}

**//13.Write a program to find the Max number from the given three number using Nested If**

import 'dart:io';

void main() {

print("Enter three numbers:");

int a = int.parse(stdin.readLineSync()!);

int b = int.parse(stdin.readLineSync()!);

int c = int.parse(stdin.readLineSync()!);

int maxNumber;

if (a >= b) {

if (a >= c) {

maxNumber = a;

} else {

maxNumber = c;

}

} else {

if (b >= c) {

maxNumber = b;

} else {

maxNumber = c;

}

}

print("The maximum number among $a, $b, and $c is: $maxNumber");

}

**//14. Write a program to find the Max number from the given three number using Ternary Operator**

import 'dart:io';

void main() {

// Prompt the user to enter three numbers

print("Enter three numbers:");

int a = int.parse(stdin.readLineSync()!);

int b = int.parse(stdin.readLineSync()!);

int c = int.parse(stdin.readLineSync()!);

int maxNumber = (a >= b && a >= c) ? a : (b >= c ? b : c);

print("The maximum number among $a, $b, and $c is: $maxNumber");

}

**//15. Write a program to find the Max number from the given three number using Nested If**

import 'dart:io';

void main() {

print("Enter three numbers:");

int a = int.parse(stdin.readLineSync()!);

int b = int.parse(stdin.readLineSync()!);

int c = int.parse(stdin.readLineSync()!);

int maxNumber;

if (a >= b) {

if (a >= c) {

maxNumber = a;

} else {

maxNumber = c;

}

} else {

if (b >= c) {

maxNumber = b;

} else {

maxNumber = c;

}

}

print("The maximum number among $a, $b, and $c is: $maxNumber");

}

**//\*16.Write a program user enter the 5 subjects mark. You have to make a**

**total and find the percentage. percentage > 75 you have to print**

**“Distinction” percentage > 60 and percentage <= 75 you have to**

**print “First class” percentage >50 and percentage <= 60 you have to**

**print “Second class” percentage > 35 and percentage <= 50 you have to**

**print “Pass class” Otherwise print “Fail”\*/**

import 'dart:io';

void main() {

List<int> marks = [];

int total = 0;

double percentage = 0.0;

print("Enter marks for 5 subjects:");

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

print("Enter marks for subject ${i + 1}:");

int mark = int.parse(stdin.readLineSync()!);

marks.add(mark);

total += mark;

}

percentage = (total / (5 \* 100)) \* 100;

String grade;

if (percentage > 75) {

grade = "Distinction";

} else if (percentage > 60) {

grade = "First class";

} else if (percentage > 50) {

grade = "Second class";

} else if (percentage > 35) {

grade = "Pass class";

} else {

grade = "Fail";

}

print("Total marks: $total");

print("Percentage: $percentage%");

print("Grade: $grade");

}

**//17. Write Program use switch statement. Display Monday to Sunday**

void main() {

int day = 6;

switch (day) {

case 1:

print("Monday");

break;

case 2:

print("Tuesday");

break;

case 3:

print("Wednesday");

break;

case 4:

print("Thursday");

break;

case 5:

print("Friday");

break;

case 6:

print("Saturday");

break;

case 7:

print("Sunday");

break;

default:

print("Invalid day");

}

}

**//18. Write a Program of Addition, Subtraction ,Multiplication and Division using Switch case.(Must Be Menu Driven)**

import 'dart:io';

void main() {

while (true) {

print("\Menu:");

print("1. Addition");

print("2. Subtraction");

print("3. Multiplication");

print("4. Division");

print("5. Exit");

print("Enter your choice:");

int choice = int.parse(stdin.readLineSync()!);

double num1, num2, result;

switch (choice) {

case 1:

print("Enter first number:");

num1 = double.parse(stdin.readLineSync()!);

print("Enter second number:");

num2 = double.parse(stdin.readLineSync()!);

result = num1 + num2;

print("Result: $result");

break;

case 2:

print("Enter first number:");

num1 = double.parse(stdin.readLineSync()!);

print("Enter second number:");

num2 = double.parse(stdin.readLineSync()!);

result = num1 - num2;

print("Result: $result");

break;

case 3:

print("Enter first number:");

num1 = double.parse(stdin.readLineSync()!);

print("Enter second number:");

num2 = double.parse(stdin.readLineSync()!);

result = num1 \* num2;

print("Result: $result");

break;

case 4:

print("Enter dividend:");

num1 = double.parse(stdin.readLineSync()!);

print("Enter divisor:");

num2 = double.parse(stdin.readLineSync()!);

if (num2 != 0) {

result = num1 / num2;

print("Result: $result");

} else {

print("Error: Division by zero!");

}

break;

case 5:

print("Exiting the program...");

return;

default:

print("Invalid choice! Please enter a valid option (1-5).");

}

}

}

**//19. Write a program of to find out the Area of Triangle, Rectangle and Circle using If Condition.(Must Be Menu Driven)**

import 'dart:io';

import 'dart:math';

void main() {

while (true) {

print("\nMenu:");

print("1. Calculate Area of Triangle");

print("2. Calculate Area of Rectangle");

print("3. Calculate Area of Circle");

print("4. Exit");

print("Enter your choice:");

int choice = int.parse(stdin.readLineSync()!);

double base, height, length, width, radius, area;

switch (choice) {

case 1:

print("Enter base of the triangle:");

base = double.parse(stdin.readLineSync()!);

print("Enter height of the triangle:");

height = double.parse(stdin.readLineSync()!);

area = 0.5 \* base \* height;

print("Area of the triangle: $area square units");

break;

case 2:

print("Enter length of the rectangle:");

length = double.parse(stdin.readLineSync()!);

print("Enter width of the rectangle:");

width = double.parse(stdin.readLineSync()!);

area = length \* width;

print("Area of the rectangle: $area square units");

break;

case 3:

print("Enter radius of the circle:");

radius = double.parse(stdin.readLineSync()!);

area = pi \* radius \* radius;

print("Area of the circle: $area square units");

break;

case 4:

print("Exiting the program...");

return;

default:

print("Invalid choice! Please enter a valid option (1-4).");

}

}

}

**//20.1 Looping Programs**

**//Write a program to print the 1 to 10 using For loop**

void main() {

for (int i = 1; i <= 10; i++) {

print(i);

}

}

**//20.2.Write a Program to print the 51 to 60 using while loop**

void main() {

int number = 51;

while (number <= 60) {

print(number);

number++;

}

}

**//20.3. Write a program to print the 100 to 81 using do while loop**

void main() {

int number = 100;

do {

print(number);

number--;

} while (number >= 81);

}

**//20.4.Write a program you have to find the factorial of given number.**

import 'dart:io';

void main() {

print("Enter a number to find its factorial:");

int number = int.parse(stdin.readLineSync()!);

int factorial = calculateFactorial(number);

print("Factorial of $number is $factorial");

}

int calculateFactorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* calculateFactorial(n - 1);

}

}

**//20.5. Write a program you have to print the Fibonacci series up to user given number**

import 'dart:io';

void main() {

print("Enter the number of terms in the Fibonacci series:");

int n = int.parse(stdin.readLineSync()!);

print("\nFibonacci Series:");

fibonacciSeries(n);

}

void fibonacciSeries(int n) {

int a = 0, b = 1;

if (n >= 1) {

print(a);

}

if (n >= 2) {

print(b);

}

for (int i = 3; i <= n; i++) {

int c = a + b;

print(c);

a = b;

b = c;

}

}

**//20.6. Write a program you have to print the table of given number.**

import 'dart:io';

void main() {

print("Enter a number to multiplication table:");

int number = int.parse(stdin.readLineSync()!);

print("\nMultiplication Table of $number:");

for (int i = 1; i <= 10; i++) {

int result = number \* i;

print("$number \* $i = $result");

}

}

**//20.7. Write a program to print the number in reverse order**

import 'dart:io';

void main() {

print("Enter a number:");

int number = int.parse(stdin.readLineSync()!);

print("\nNumber in reverse order:");

print(reverseNumber(number));

}

int reverseNumber(int n) {

int reversed = 0;

while (n != 0) {

int digit = n % 10;

reversed = reversed \* 10 + digit;

n ~/= 10;

}

return reversed;

}

**//20.8.Write a program to find out the max from given number (E.g. No: -1562 Max number is 6**

import 'dart:io';

void main() {

print("Enter a number:");

String input = stdin.readLineSync()!;

int maxDigit = findMaxDigit(input);

print("Max number is $maxDigit");

}

int findMaxDigit(String input) {

List<String> digits = input.replaceAll('-', '').split('');

int maxDigit = 0;

for (String digit in digits) {

int currentDigit = int.parse(digit);

if (currentDigit > maxDigit) {

maxDigit = currentDigit;

}

}

return maxDigit;

}

**//20.9 Write a program make a summation of given number(E.g. 1523 ans :-11)**

import 'dart:io';

void main() {

print("Enter a number:");

String input = stdin.readLineSync()!;

int number = int.parse(input);

int sum = sumOfDigits(number.abs());

if (number < 0) {

sum \*= -1;

}

print("Sum of digits is: $sum");

}

int sumOfDigits(int num) {

int sum = 0;

while (num != 0) {

sum += num % 10;

num ~/= 10;

}

return sum;

}

**//20.10.Write a program you have to make a summation of first and last Digit.(E.g. 1234 ans:-5)**

import 'dart:io';

void main() {

print("Enter a number:");

String input = stdin.readLineSync()!;

int number = int.parse(input);

int sum = firstAndLastDigitSum(number);

print("Sum of first and last digits is: $sum");

}

int firstAndLastDigitSum(int num) {

int lastDigit = num % 10;

int firstDigit = num;

while (firstDigit >= 10) {

firstDigit ~/= 10;

}

return firstDigit + lastDigit;

}

**21. what is inharitance**

ans. child class derives properties and behavior from and existing parent class is called inharitance.

**22. Which inheritance is not supported by Dart? Why? B3. What is advantage of inheritance?**

ans. Multiple Inheritance is not supported by Dart.

**advance inharitance :-**

**1.Code Reusability:**

Inheritance allows a new class (child class) to reuse code from an existing class (parent class). This reduces redundancy and leads to a more modular codebase.

**2.Simplified Maintenance:**

Changes made to the parent class automatically propagate to child classes.

**3.Extensibility:**

New functionality can be added to existing code by creating new child classes.

**4. Polymorphism**

Inheritance supports polymorphism, allowing a single function to operate on objects of different classes.

**5.Improved Code Organization:**

It helps organize code into a hierarchical structure, making it more understandable and logical.

**6.Abstraction:**

Inheritance helps achieve a higher level of abstraction.

**23.Difference between inheritance and encapsulation.**

ans.

Definition:

Encapsulation is the practice of bundling the data (attributes) and methods (functions) that operate on the data into a single unit, typically a class. It restricts direct access to some of an object's components, which can prevent the accidental modification of data.

Inheritance is a mechanism where a new class (child class) is derived from an existing class (parent class). The child class inherits the attributes and methods of the parent class, allowing for code reuse and the creation of a hierarchical relationship between classes.

Purpose:

Facilitates code reuse and establishes a relationship between classes, where the child class can inherit and extend the functionality of the parent class.

Enhances data security and integrity by controlling access to the data through access modifiers (e.g., private, protected, public). Promotes modularity and helps in hiding the internal implementation details of a class from the outside world.

Implementation:

Achieved by defining a new class that extends an existing class. In Dart, this is done using the extends keyword.

Achieved by defining private fields and providing public getter and setter methods. In Dart, an underscore (\_) is used to mark a field as private.

B5. Difference between inheritance and abstraction.

ans.

Purpose:

Inheritance: Primarily used for code reuse and establishing hierarchical relationships between classes.

Abstraction: Used to hide complexity and show only the essential features of an object.

Implementation:

Inheritance: Implemented using the extends keyword to create a child class that inherits from a parent class.

Abstraction: Implemented using abstract classes and interfaces to define a clear and simplified interface.

Usage:

Inheritance: When you need to create a new class based on an existing class and extend its functionality.

Abstraction: When you need to define a simplified interface and hide the complex implementation details.

Focus:

Inheritance: Focuses on creating a relationship between parent and child classes.

Abstraction: Focuses on exposing only the necessary parts of an object while hiding the internal details.

**24. Difference between inheritance and polymorphism**

ans.

Purpose:

Inheritance: Used for creating a hierarchical relationship and promoting code reuse by extending the functionality of existing classes.

Polymorphism: Used for allowing objects of different types to be treated as objects of a common super type, promoting flexibility and integration.

Focus:

Inheritance: Focuses on the relationship between parent and child classes.

Polymorphism: Focuses on the ability to use a single interface to represent different underlying forms (data types).

Implementation:

Inheritance: Implemented using the extends keyword to create child classes that inherit from parent classes.

Polymorphism: Implemented through method overriding (runtime polymorphism) and method overloading (compile-time polymorphism).

**25. Can we override static method in Dart?**

ans. In Dart, cannot override static methods. Static methods belong to the class itself rather than to any instance of the class. Because of this, they are not part of the instance's method set and cannot be overridden in subclasses.

**26. Can we overload static method in Dart?**

ans. Dart does not support method overloading

**27. Can a class implement more than one interface? B10. Can a class extend more than one class in Dart?**

Ans. Yes, a class in Dart can implement multiple interfaces.and

No, a class in Dart cannot extend more than one class. Dart does not support multiple inheritance

**28. Can an Interface Extend More Than One Interface in Dart?**

Yes, in Dart, an interface (which is essentially an abstract class) can extend more than one interface (abstract class).

**29. What will happen if a class implements two interfaces and they both have a method with same name and signature?**

the class is required to provide a single implementation of that method. Here is how it works, with an example:

interface InterfaceA {

void commonMethod();

}

interface InterfaceB {

void commonMethod();

}

public class MyClass implements InterfaceA, InterfaceB {

// Provide the implementation for the commonMethod

@Override

public void commonMethod() {

// Implementation of the method

System.out.println("commonMethod implementation");

}

public static void main(String[] args) {

MyClass myObject = new MyClass();

myObject.commonMethod();

}

}

**30.Can we pass an object of a subclass to a method expecting an object of the super class? B14.Are static members inherited to sub classes?**

Yes, you can pass an object of a subclass to a method that expects an object of the superclass. This is because of a concept known as polymorphism. Static members are not inherited in the same way that instance members are. However, they can be accessed using the subclass name.

class Parent {

static void staticMethod() {

System.out.println("Static method in Parent");

}

}

class Child extends Parent {

static void staticMethod() {

System.out.println("Static method in Child");

}

}

public class MainClass {

public static void main(String[] args) {

Parent.staticMethod(); // Output: Static method in Parent

Child.staticMethod(); // Output: Static method in Child

}

}

**31. What happens if the parent and the child class have a field with same identifier? B16. Are constructors and initializers also inherited to sub classes?**

class Parent {

String field = "Parent Field";

}

class Child extends Parent {

String field = "Child Field";

void printFields() {

System.out.println(field);

System.out.println(super.field);

}

}

public class MainClass {

public static void main(String[] args) {

Child child = new Child();

child.printFields();

}

}

**32.How do you restrict a member of a class from inheriting by its sub classes?**

Ans. To restrict a member of a class (such as a field or method) from being inherited by its subclasses, you can use access modifiers.

class Parent {

private void privateMethod() {

System.out.println("This is a private method in Parent");

}

public void publicMethod() {

System.out.println("This is a public method in Parent");

}

}

class Child extends Parent {

public void accessMethods() {

publicMethod();

}

}

public class MainClass {

public static void main(String[] args) {

Child child = new Child();

child.accessMethods();

}

}

**33.How do you implement multiple inheritance in Dart?**

In Dart, multiple inheritance is not directly supported, meaning a class cannot inherit from more than one superclass. However, you can achieve similar functionality using interfaces.

**34.Can a class extend by itself in Dart?**

In Dart, multiple inheritance is not directly supported, meaning a class cannot inherit from more than one superclass. However, you can achieve similar functionality using interfaces.

**35. How do you override a private method in Dart?**

Ans. In Dart, private methods are denoted by an underscore (\_) at the beginning of the method name. When a method is marked as private, it can only be accessed within the file (library) where it is defined. Therefore, you cannot directly override a private method from a superclass in a subclass if the subclass is in a different file.

**36.When to overload a method in Dart and when to override it?**

Method Overloading

Definition: Method overloading allows you to define multiple methods with the same name but different parameter lists (types, number, or order of parameters) within the same class.

When to Use:

Different Parameter Types: When you want to provide different ways to invoke a method based on the type of arguments passed.

Varied Number of Parameters: When you want to allow flexibility in how many arguments are provided.

Improving Readability: To enhance the readability of your code by using a consistent method name while varying the method's behavior based on input.

class Calculator {

// Overloaded method with different parameter types

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

// Overloaded method with a different number of parameters

int addMultiple(int a, int b, [int c = 0]) {

return a + b + c;

}

}

void main() {

Calculator calculator = Calculator();

print(calculator.add(5, 10)); // Output: 15

print(calculator.add(5.5, 10.5)); // Output: 16.0

print(calculator.addMultiple(5, 10)); // Output: 15

print(calculator.addMultiple(5, 10, 15)); // Output: 30

}

Method Overriding

Definition: Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass.

When to Use:

Customization of Behavior: When you want to change or extend the behavior of a method from a superclass in a subclass.

Polymorphism: To allow different subclasses to have their own implementations of the same method, enabling dynamic method resolution at runtime.

When using Inheritance: When working with inheritance, you might need to override methods to adapt inherited behavior to suit the subclass.

class Animal {

void speak() {

print('Animal makes a sound');

}

}

class Dog extends Animal {

@override

void speak() {

print('Dog barks');

}

}

class Cat extends Animal {

@override

void speak() {

print('Cat meows');

}

}

void main() {

Animal myDog = Dog();

Animal myCat = Cat();

myDog.speak(); // Output: Dog barks

myCat.speak(); // Output: Cat meows

}

**37. What the order is of extends and implements keyword on Dart class declaration?**

Ans. extends: This keyword is used to indicate that the class is inheriting from a superclass. You can only extend one class (single inheritance).

implements: This keyword is used to implement one or more interfaces. A class can implement multiple interfaces.

class Animal {

void makeSound() {

System.out.println("Animal makes a sound");

}

}

interface CanFly {

void fly();

}

interface CanSwim {

void swim();

}

public class Duck extends Animal implements CanFly, CanSwim {

@Override

void makeSound() {

System.out.println("Duck quacks");

}

@Override

public void fly() {

System.out.println("Duck flies");

}

@Override

public void swim() {

System.out.println("Duck swims");

}

public static void main(String[] args) {

Duck duck = new Duck();

duck.makeSound(); // Output: Duck quacks

duck.fly(); // Output: Duck flies

duck.swim(); // Output: Duck swims

}

}

**38. How do you prevent overriding a Dart method without using the final modifier?**

in Dart, to prevent a method from being overridden in subclasses without using the final modifier, you can use the @override annotation in combination with making the method private. Private methods in Dart are denoted by starting the method name with an underscore (\_).

**39. What are the rules of method overriding in Dart?**

A. Use the same name and parameters for the method in the subclass.

B. It’s best to use @override to show that you are overriding a method.

C. The return type can be the same or a subtype.

D. Don’t make the access level more restrictive.

E. Final methods cannot be overridden.

F. Static methods are hidden, not overridden.

**40. Difference between method overriding and overloading in Dart.**

• Method Overloading

A. Multiple methods of same name in single class.

B. No need of inheritance, as it is in single class.

C. All methods have different signature.

D. It's a compile time polymorphism.

E. No special keyword used.

Method Overriding

A. Multiple methods of same name in different class.

B. Inheritance is used, as it is in different class.

C. All methods have same signature.

D. It's a run time polymorphism.

E. Virtual & override keyword use

**41.What happens when a class implements two interfaces and both declare field (variable) with same name?**

ans. When a class implements multiple interfaces that declare fields with the same name, it results in a compile-time error due to name conflicts. To avoid this, prefer using getter and setter methods in interfaces, or ensure that fields have unique names.

**42.Can a subclass instance method override a superclass static method?**

ans. No, a subclass instance method cannot override a superclass static method in Dart

**43.Can a subclass static method hide superclass instance method?**

No, a subclass static method cannot hide a superclass instance method in Dart.

**44.Can a superclass access subclass member?**

No, a superclass cannot directly access the members (fields or methods) of a subclass in Dart.

**45. Difference between object oriented and object based language.**

Object-Oriented Languages

Definition: Based on the concept of objects, supporting OOP principles.

Core Principles : Encapsulation, inheritance, and polymorphism.

Examples: Java, C++, Python, C#, Ruby.

Support for Classes : Requires classes for creating objects.

Object-Based Languages :

Definition:Support objects but not full OOP principles.

Core Principles : Primarily encapsulation; lacks inheritance and polymorphism.

Examples: JavaScript (before ES6), VBScript, some versions of Pascal.

Support for Classes : May not require classes; objects can be created independently.

**46. Create a program using List**

import 'package:flutter/material.dart';

import 'package:flutter\_application\_1/student.dart';

void main() {

// growable this value it can change

List growableList = [];

growableList.add(new Student(name: "S1", age: 25, markes: 50));

growableList.add(new Student(name: "S2", age: 28, markes: 32));

growableList.add(new Student(name: "S3", age: 32, markes: 40));

growableList.add(new Student(name: "S4", age: 22, markes: 62));

growableList.add(new Student(name: "S5", age: 20, markes: 46));

runApp(MaterialApp(

home: Scaffold(

body: ListView.builder(

itemCount: growableList.length,

itemBuilder: (context, index) => Card(

color: growableList[index].markes > 50

? Colors.blueGrey

: Colors.redAccent,

elevation: 10,

child: Padding(

padding: EdgeInsets.all(16),

child: Text(

"Student Name is ${growableList[index].name.toString()}",

style: TextStyle(fontSize: 20),

)),

),

),

),

));

}

**47. Create a program using Set**

void main() {

// Creating a set of integers

Set<int> numbers = {10, 20, 30, 40, 50};

// Printing the original set

print('Original Set: $numbers');

// Adding an element to the set

numbers.add(60);

print('After adding 60: $numbers');

// Attempting to add a duplicate element

numbers.add(20); // This will not be added

print('After trying to add 20 again: $numbers');

// Removing an element from the set

numbers.remove(30);

print('After removing 30: $numbers');

// Checking if an element exists in the set

bool has40 = numbers.contains(40);

print('Does the set contain 40? $has40');

// Iterating through the set

print('All elements in the set:');

for (int number in numbers) {

print(number);

}

}

**48. Create a program using Map**

void main() {

// Creating a map with string keys and integer values

Map<String, int> ages = {

'Alice': 30,

'joy': 25,

'Charlie': 35,

};

// Printing the original map

print('Original Map: $ages');

// Adding a new key-value pair to the map

ages['David'] = 40;

print('After adding David: $ages');

// Removing a key-value pair from the map

ages.remove('joy');

print('After removing joy: $ages');

// Accessing a value by key

int aliceAge = ages['Alice']!;

print('Alice\'s age: $aliceAge');

// Checking if a key exists in the map

bool hasCharlie = ages.containsKey('Charlie');

print('Does the map contain Charlie? $hasCharlie');

// Iterating through the map

print('All entries in the map:');

ages.forEach((key, value) {

print('$key: $value');

});

}