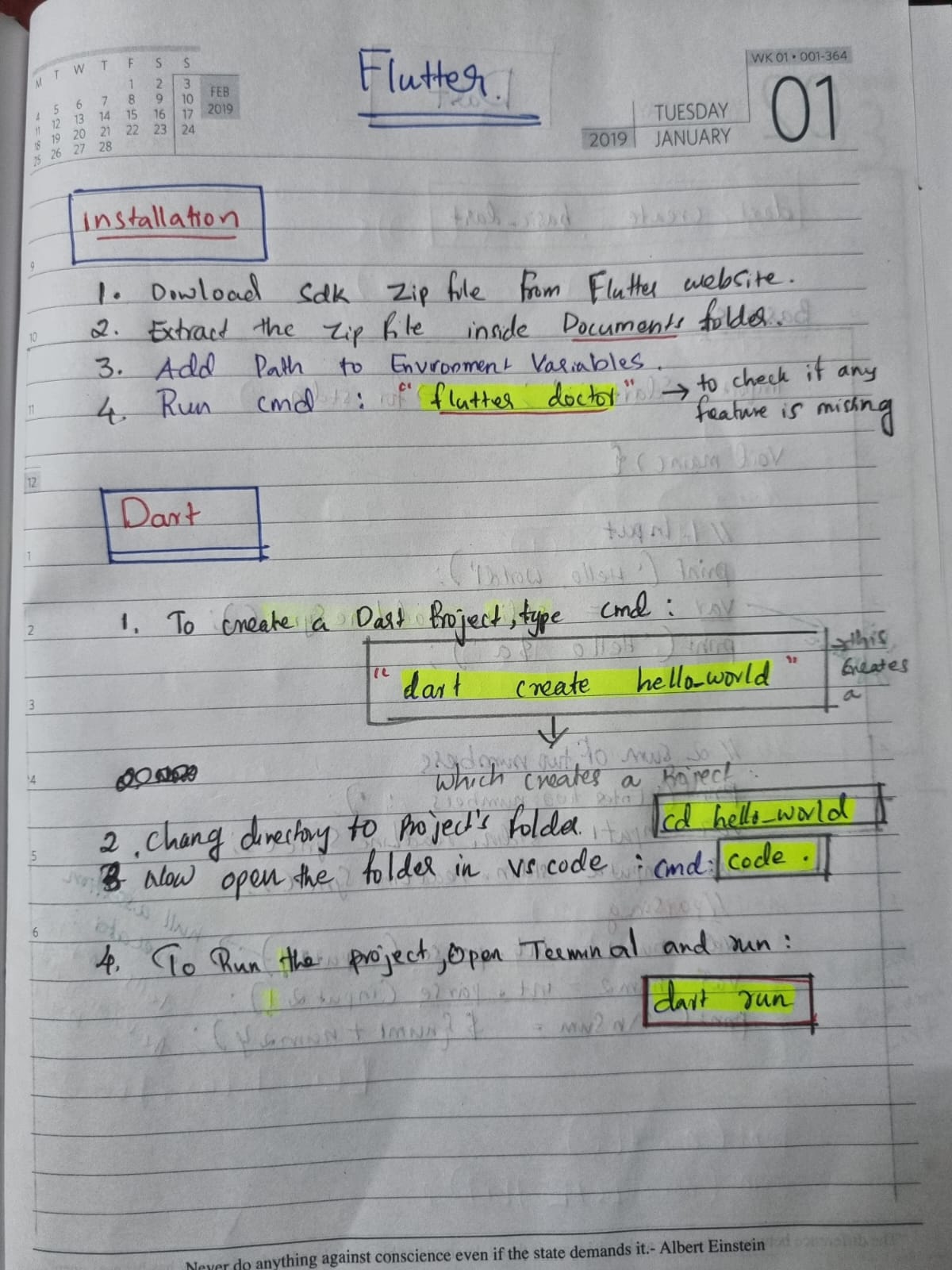
**-DART-**

Dart is an object-oriented language with C-style syntax which can optionally trans compile into JavaScript. It supports a varied range of programming aids like interfaces, classes, collections, generics, and optional typing.



# 1.BASICS

|  |
| --- |
| import 'package:p1\_basics/p1\_basics.dart' as p1\_basics;  import 'dart:io'; //for stdin  //BASICS  void main(){    //1.Input    print('Hello World!');    var a = stdin.readLineSync();    print('Hello $a');      //2.Sum of two Numbers    print('\nEnter two numbers : ');    var input1 = stdin.readLineSync();    var input2 = stdin.readLineSync();    //PARSING...    var num1 = int.parse(input1!); //null assertion operator : to tell the Dart compiler the value is not null.    var num2 = int.parse(input2!);    print('\nSum = ${num1 + num2}');    //3.DataTypes    int x = 10;    double y = 20.5;    //'num' datatype can store both int and double values    num n1 = 10;    num n2 = 20.5;    String myname = 'Anshad';    String details = " I'am Anshad ";    String details2 = " I\"m Anshad";  // \" escapes the double quote, so the string remains valid.    //Multiline String:    String address = '''house number 123    wayanad''';    print(myname.length); |

**Output**

|  |
| --- |
| PS E:\MCA\COURSES\Flutter\DART\_programs\p1\_basics> dart run  Building package executable...  Built p1\_basics:p1\_basics.  Hello World!  Anshad  Hello Anshad  Enter two numbers :  3  4  Sum = 7  6  PS E:\MCA\COURSES\Flutter\DART\_programs\p1\_basics> |

**LIST-Array**

|  |
| --- |
| //4.List - act as ARRAY      List<int> numberList = [55,24,17,62]; //List    List<int> listToAdd = [0,1,2];      var dynamic\_list = ['Anshad',24,81.52];  //Dynamic List      //4.1:Find length of List    print(numberList.length);    //4.2:To find Element in a List use 'contains()' property :    if(numberList.contains(55)){      print('List contains 55');    }    else{      print('List doesnt contains 55');    }    //4.3:To Add an element to List :    numberList.add(90);    //4.4:To Remove an element from List :    numberList.removeAt(0); //remove element by index    numberList.removeLast(); //removes last element of list    //4.5: To Add list of elements to a list    numberList.addAll(listToAdd);    //4.6: Print    print(numberList);    print(numberList.join( " - ")); //Print uing seperator    //4.7 : Nested List    List<List<int>> nestedList = [      [1,2,3],      [4,5,6],      [7,8,9]    ];    print(nestedList);    print(nestedList[0][1]); |

Output

|  |
| --- |
| 4  List contains 55  [24, 17, 62, 0, 1, 2]  24 - 17 - 62 - 0 - 1 - 2  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  2  PS E:\MCA\COURSES\Flutter\DART\_programs\p1\_basics> |

**LIST vs SET**

|  |
| --- |
| //5.LIST VS SET    List<int> numberList2 = [1,1,1,2,2,3,5,8,8,8,9];    Set<int> numberSet = {1,1,1,2,2,3,5,8,8,8,9}; //SET does not support Duplicate values    print(numberList2);    print(numberSet); |

**Output:**

|  |
| --- |
| **[1, 1, 1, 2, 2, 3, 5, 8, 8, 8, 9]**  **{1, 2, 3, 5, 8, 9}** |

**Map - Dictionary**

|  |
| --- |
| //6.MAP - act as Dictionary    Map<String,String> map1 = {      "name" : "Anshad",      "place" : "wayanad"    };    var dynamicMap = {      "name" : "Anshad Muhammad",      "age" : 24    };    print(map1["name"]);    print(dynamicMap["age"]); |

**Output:**

|  |
| --- |
| **Anshad**  **24** |

# Functions and Named Parameters

|  |
| --- |
| import 'package:part3\_dartfunctions/part3\_dartfunctions.dart' as part3\_dartfunctions;  import 'dart:io';  //Function and Named Parameters(Required parameter,Option Parameter and Default Parameter)  void main(){    //1.1 : Function Without Return value and No Parameters    sum();    //1.2 : Function Without Return value ,But have Parameters    sumParams(5,7);    //1.3 : Function with Return value and Parameters    var sum1 = sumReturns(10, 15);    print(sum1);    //2.1 : Named Parameters : Required Parameter ,Option Parameter and Default parameters    sumReq(firstNumber : 100 , secondNumber :200 , third : 34);    //2.2  : Passing Function as Parameter - We can write this in Two ways:      // sumFunction(23 ,24 , sumParams);    //OR Using ANONYMOUS FUNCTION :    sumFunction(23 ,24 , (int f , int s){      print('Function sum = ${f+s}');    });    //3.1 : FUTURE FUNCTION    // sumFuture(12 , 12);    // print('After sumFuture');    sum2();    print('After future');  }  void sum(){    print(2+3);  }  void sumParams(int a , int b){    print('${a+b}');  }  int sumReturns(int a , int b){    return a+b;  }  void sumReq({required int firstNumber,required int secondNumber , int third=0}){    print(firstNumber + secondNumber + third) ;  }  //passed function as parameter:  void sumFunction(int a , int b ,void Function(int,int) customSum){    customSum(a,b);  }  //FUTURE FUNCTION :  Future<int> sumFuture(int a ,int b) async{    await Future.delayed(Duration(seconds: 3)); //use await -To show the implementation of Future function    //so that 3 seconds delayed before printing sum.    print('In Sum Future = ${a+b}');    return a+b;  }  Future<void> sum2()async{    await sumFuture(33, 44); //await can be used in future function    print('In just Sum');  } |

**Output :**

|  |
| --- |
| **PS E:\MCA\COURSES\Flutter\DART\_programs\part3\_dartfunctions> dart run**  **Building package executable...**  **Built part3\_dartfunctions:part3\_dartfunctions.**  **5**  **12**  **25**  **334**  **Function sum = 47**  **After future**  **In Sum Future = 77**  **In just Sum**  **PS E:\MCA\COURSES\Flutter\DART\_programs\part3\_dartfunctions>** |

## **FUTURE FUNCTION**

|  |
| --- |
| import 'dart:io';  //Using Main function as FUTURE FUNCTION:  Future<void> main() async{    await sum2();    print('After sum');  }  //FUTURE FUNCTION :  Future<int> sumFuture(int a ,int b) async{    await Future.delayed(Duration(seconds: 3)); //use await -To show the implementation of Future function    //so that 3 seconds delayed before printing sum.    print('In Sum Future = ${a+b}');    return a+b;  }  Future<void> sum2()async{    await sumFuture(33, 44); //await can be used in future function    print('In just Sum');  } |

**Output:**

|  |
| --- |
| **In Sum Future = 77**  **In just Sum**  **After sum**  **PS E:\MCA\COURSES\Flutter\DART\_programs\part3\_dartfunctions>** |

# Class and Object :

## **Const VS Final**

|  |
| --- |
| import 'package:part4\_class\_object/part4\_class\_object.dart' as part4\_class\_object;  void main(List<String> arguments) {    //Final VS Const    //1."final" keyword    final String name;    name = "Anshad";    //name = "Muhammad"; //This give Error , since "final" keyword cannot be altered in future.    final List<int> numberList;    numberList = List.empty(); //To empty the List    numberList.add(10);    numberList.add(20);    //numberList =[10,20]; //Gives Error , since even list when set as final cannot be changed.    //2."const" -  const makes the variable constant from compile-time only.    //const String name2;    //name2 = "Anshad2";    //So we cannot decalre a const values like this.Instead we have to initialize while declaring itself.    const String name2 = "Anshad2";  } |

## **Class and Object:**

|  |
| --- |
| //1.Class :  class Person{    String? name; //? - for null safety    int age = 24;    //Constructor:    Person(String name , int age){      this.name = name;      this.age = age;    }  }  void main(List<String> arguments) {    //2.Object :    final person = Person('Anshad',24);    person.name = "Muhammad";      print({person.name,person.age});  } |

|  |
| --- |
| {Muhammad, 24} |

## **Constructor –implementation in Different way:**

|  |
| --- |
| //1.Class :  class Person{    final String name;    final int age;    //Constructor - in a differnt way    Person(this.name , this.age);  }  void main(List<String> arguments) {    //2.Object :    final person = Person('Anshad',24);    // person.name = "Muhammad"; //Now this gives Error, Bcs name is decalred as "final"    print({person.name,person.age});  } |

## **Named Constructors:**

|  |
| --- |
| //1.Class :  class Person{    final String name;    final int age;    //Constructor - in a differnt way    Person(this.name , this.age);    Person.ageBelow50(this.name , this.age){      print(name);    }    Person.ageAbove50(this.name,this.age){      print(name);    }  }  void main(List<String> arguments) {    //2.Object :    // final person = Person('Anshad',24);    // person.name = "Muhammad"; //Now this gives Error, Bcs name is decalred as "final"      // print({person.name,person.age});    final person = Person.ageAbove50("Anshad",24);    print(person.name);    print(person.age);  } |

|  |
| --- |
| Anshad  Anshad  24 |

## **Functions – Comparing with Constructors:**

|  |
| --- |
| //1.Class :  class Person{    final String name;    final int age;    //Constructor - in a differnt way    Person(this.name , this.age);    Person.ageBelow50(this.name , this.age){      print(name);    }    Person.ageAbove50(this.name,this.age){      print(name);    }    //function    void sayHai(){      print("Hi!");    }  }  void main(List<String> arguments) {    //2.Object :    // final person = Person('Anshad',24);    // person.name = "Muhammad"; //Now this gives Error, Bcs name is decalred as "final"      // print({person.name,person.age});    final person = Person.ageAbove50("Anshad",24);    print(person.name);    print(person.age);    //Accessing functions inside a class using objects    person.sayHai();  } |

**.**

**NOTE: To make any variable private – Add underscore at beginning!**

|  |
| --- |
| //1.Class :  class Person{    final String \_name; //Add (\_) to make it private.    final int \_age;    //Constructor - in a differnt way    Person(this.\_name , this.\_age);    Person.ageBelow50(this.\_name , this.\_age){      print(\_name);    }    Person.ageAbove50(this.\_name,this.\_age){      print(\_name);    }    //function    void sayHai(){      print("Hi!");    }  }  void main(List<String> arguments) {    //2.Object :    final person = Person.ageAbove50("Anshad",24);    print(person.\_name); //Here we are able to access the private variable bcs in Dart we can access private varibles of same file    print(person.\_age);    //Accessing functions inside a class using objects    person.sayHai();  } |

## **How to Access a Private variables of another file?:**

**Solution : Use Getter and Setter.**

**First lets try function to get and set values:**

* **File:Person.dart**

|  |
| --- |
| //1.Class :  class Person{    final String \_name; //Add (\_) to make it private.    final int \_age;    //Constructor - in a differnt way    Person(this.\_name , this.\_age);    Person.ageBelow50(this.\_name , this.\_age){      print(\_name);    }    Person.ageAbove50(this.\_name,this.\_age){      print(\_name);    }      //function    void sayHai(){      print("Hi!");    }    //Act as getter - To Pass private-age to main file    int getAgeFunction(){      return \_age;    }    String getNameFunction(){      return \_name;    }  } |

* **File:main.dart**

|  |
| --- |
| //Import class - Person  import 'person.dart';  void main(List<String> arguments) {    //2.Object :    // final person = Person('Anshad',24);    // person.name = "Muhammad"; //Now this gives Error, Bcs name is decalred as "final"      // print({person.name,person.age});    final person = Person.ageAbove50("Anshad",24);    //Accessong private variables from other files:    print(person.getNameFunction());    print(person.getAgeFunction());    //Accessing functions inside a class using objects    person.sayHai();  } |

**Now Use get and set :**

* **File:Person.dart**

|  |
| --- |
| //1.Class :  class Person{    final String \_name; //Add (\_) to make it private.    int \_age; //removed final to access it in setter.    //Constructor - in a differnt way    Person(this.\_name , this.\_age);    Person.ageBelow50(this.\_name , this.\_age){      print(\_name);    }    Person.ageAbove50(this.\_name,this.\_age){      print(\_name);    }      //function    void sayHai(){      print("Hi!");    }    //Act as getter - To Pass private-age to main file    String getNameFunction(){      return \_name;    }    int getAgeFunction(){      return \_age;    }      //Getter - get    String get getName{      return \_name;    }    int get getAge{      return \_age;    }    //Setter - set    set setAge(int value){      \_age = value;    }  } |

* **File:main.dart**

|  |
| --- |
| //Import class - Person  import 'person.dart';  void main(List<String> arguments) {    //2.Object :    // final person = Person('Anshad',24);    // person.name = "Muhammad"; //Now this gives Error, Bcs name is decalred as "final"      // print({person.name,person.age});    final person = Person.ageAbove50("Anshad",24);    //Accessing private variables from other files using functions:    print(person.getNameFunction());    print(person.getAgeFunction());    //Accessing private variables from other files using "get"-getter:    print(person.getName);    print(person.getAge);    //Setting values using "set" - setter    person.setAge = 30 ;    //Accessing functions inside a class using objects    person.sayHai();  } |

# Inheritance and Exceptions :

## **Inheritance – basic program**

|  |
| --- |
| //Inheritance  //Base class/Parent class  class Animal{    void sayHello(){      print("Animal say Hello!");    }  }  //child class  class Human extends Animal{    void sayName(){      print("Say name");    }  }  void main(List<String> arguments) {    //Object:    final human = Human();    //calling functions    human.sayHello(); //calling function of base class    human.sayName(); //calling function of child class  } |

|  |
| --- |
| Animal say Hello!  Say name |

## **Inheritance – @Override**

NOTE: DART doesn’t allow Multiple Inherirance ,But it can be achieved by “**MIXINS**”.

|  |
| --- |
| //Inheritance  //Base class/Parent class  class Animal{    void sayHello(){      print("Animal say Hello!");    }  }  //child class  class Human extends Animal{    void sayName(){      print("Say name");    }      @override    void sayHello() {      // TODO: implement sayHello      print("Say Hello Human!");      super.sayHello();    }  }  void main(List<String> arguments) {    //Object:    final human = Human();    //calling functions    human.sayHello(); //calling function of base class    human.sayName(); //calling function of child class  } |

|  |
| --- |
| Say Hello Human!  Animal say Hello!  Say name |

## **Inheritance – Interface Implementation using Abstract class**

NOTE: Abstract class can be used as Interface in DART.

|  |
| --- |
| //Abstract class - for Interface  abstract class Animal{    void sayHello();  }  //Interface : (implements)  class Human implements Animal{    @override //Here we use override ,bcs Human class is overriding sayHello() function of Animal class -which is an abstract class    void sayHello() {      // TODO: implement sayHello      print("Human Hello!");    }    void sayName(){      print("Name");    }  }  void main(List<String> arguments) {    //Object:    final human = Human();    //calling functions    human.sayHello(); //calling function of base class    // human.sayName(); //calling function of child class  } |

|  |
| --- |
| Human Hello! |