Basics of Dart:-

## 1. Dart:-

Dart is an open-source, general-purpose, object-oriented programming language with C-style syntax developed by Google in 2011. The purpose of Dart programming is to create a frontend user interfaces for the web and mobile apps.

It is under active development, compiled to native machine code for building mobile apps, inspired by other programming languages such as Java, JavaScript, C#, and is Strongly Typed. Since Dart is a compiled language so you cannot execute your code directly; instead, the compiler parses it and transfer it into machine code.

It supports most of the common concepts of programming languages like classes, interfaces, functions, unlike other programming languages. Dart language does not support arrays directly. It supports collection, which is used to replicate the data structure such as arrays, generics, and optional typing.

The following example shows simple Dart programming.

```
void main() {
  for (int i = 0; i < 5; i++) {
    print('hello ${i + 1}');
  }
}
```

## 2. DataType:-

A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

Dart is a Strongly Typed programming language. It means, each value you use in your programming language has a type either string or number and must be known when the code is compiled.

Here, we are going to discuss the most common basic data types used in the Dart programming language.

| DataType | Example |
|---|---|
| Num, int,<br>double | int age = 25;<br>double price = 125.50; |
| Object | Person = Person() |
| boolean | bool var_name = true;  Or bool var_name = false; |

## 3. Variables and Functions:-

In programming, a variable is a value that can change, depending on conditions or on information passed to the program.

Typically, a program consists of instructions that tell the computer what to do and data that the program uses when it is running.

Variables are the namespaces in memory that store values.

The name of a variable is called an identifier.

They are the data containers, which can store the value of any type. For example:- var myAge = 50;

Here, myAge is a variable that stores an integer value 50. We can also give it int and double. However, Dart has a feature Type Inference, which infers the types of values. So, if you create a variable with a var keyword, Dart can infer that variable as of type integer.

# Functions:-

Functions are another core feature of any programming language.

Functions are a set of statements that performs a specific task. They are organised into the logical blocks of code that are readable, maintainable, and reusable.

The function declaration contains the function name, return type, and parameters. The following example explains the function used in Dart programming.For example:-

```
//Function declaration

num addNumbers(num a, num b) {
```

```
  // Here, we use num as a type because it should work with int and double both.
  return a + b;
}
var price1 = 29.99;
var price2 = 20.81;
var total = addNumbers(price1, price2);
var num1 = 10;
var num2 = 45;
var total2 = addNumbers(num1, num2);
```

4.Operators:-Operators are the foundation of any programming language. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands.Dart language supports all operators, as you are familiar with other programming languages such as C, Kotlin, and Swift. The operator's name is listed below:-

Arithmetic

Relational

Logical

Bitwise

Assignment

Arithmetic Operators:-These operators are used to perform arithmetic/mathematical operations on operands. Examples: (+, -, *, /, %,++,–). Arithmetic operators are of two types:-

a.Unary Operator:-Operators that operate or work with a single operand are unary operators. For example: Increment(++) and Decrement(–) Operators.

b.Binary Operator:-Operators that operate or work with two operands are binary operators. For example: Addition(+), Subtraction(-), multiplication(*), Division(/) operators

2.Relational Operator:-These are used for the comparison of the values of two operands. For example, checking if one operand is equal to the other operand or not,

whether an operand is greater than the other operand or not, etc. Some of the relational operators are (==, >= , <= ).

Example:- int a=3; int b=4;

cout<<(a<b);

3.Logical Operator:-Logical Operators are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a Boolean value either true or false.

For example, the logical AND represented as the '&&' operator in C returns true when both the conditions under consideration are satisfied. Otherwise, it returns false. Therefore, a && b returns true when both a and b are true (i.e. non-zero)

eg:-cout<<((4!=5) && (4<5));

4.Bitwise Operator:-The Bitwise operators are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. Mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit level for faster processing. For example, the bitwise AND operator represented as '&' in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1(True).

Eg:- int a=5, b=9;

cout<<(a^b);

cout<<(~a);

5.Assignment Operator:-Assignment operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value. The value on the right side must be of the same data type as the variable on the left side otherwise the compiler will raise an error.

5. Decision Making and Looping:-Decision making structures require that the programmer specifies one or more conditions to be evaluated or tested by the program, along

with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.The decision-making is a feature that allows you to evaluate a condition before the instructions are executed. The Dart language supports the following types of decision-making statements:-

If statement

If-else statement

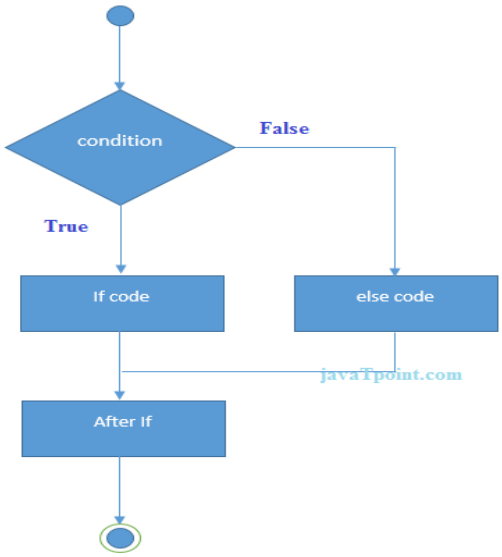
Switch statement

If statement:-An if statement consists of a boolean expression followed by one or more statements.

If-else statement:-An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

Switch Statement:-A switch statement allows a variable to be tested for equality against a list of values.

The below diagram explains it more clearly.




Loops are used to execute a block of code repeatedly until a specified condition becomes true. Dart language supports the following types of loop statements:-

for

for..each

while

Do..while

1.For:-The initialization statement is executed only once.Then, the test expression is evaluated. If the test expression is evaluated to false, the `for` loop is terminated.

Example:-`for (initializationStatement; testExpression; updateStatement)`

```
{
    // statements inside the body of loop
}
```

2.For each:-The for-each loop is used to traverse arrays or collections in Java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

Example:-

```
for(data_type variable : array_name){
//code to be executed}
```

3. While:-The while loop is used to repeat a section of code an unknown number of times until a specific condition is met. For example, say we want to know how many times a given number can be divided by 2 before it is less than or equal to 1. If we know a specific number, such as 32, we can say 5 times, but for a given symbolic variable "NUMBER" which represents any number in the world, how many times is not known a priori (beforehand).

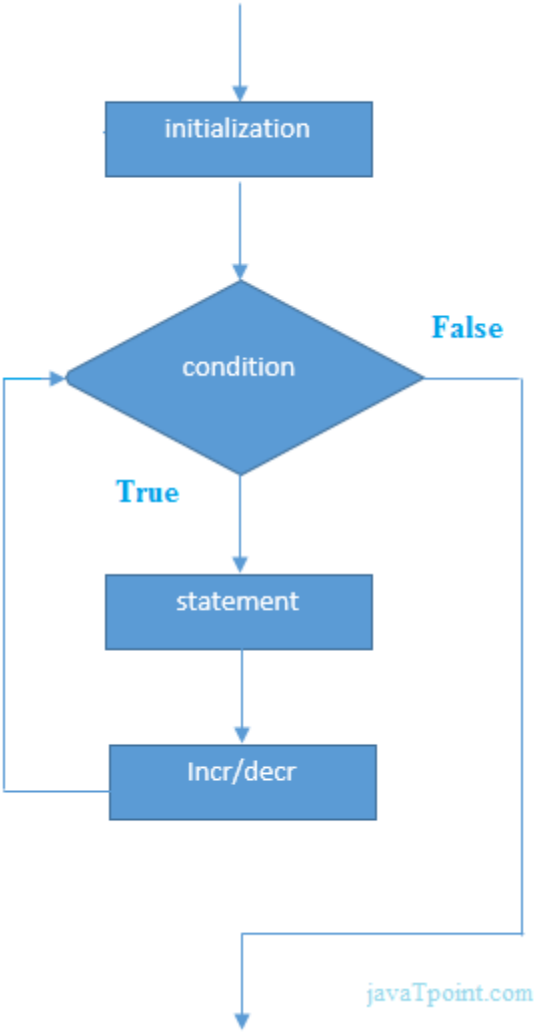
```
// Print numbers from 1 to 5


#include <stdio.h>
int main() {
 int i = 1;
  while (i <= 5) {
   printf("%d\n", i);
   ++i;
 }return 0;
}
```

4.Do While:-A do while loop is a control flow statement that executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given boolean condition at the end of the block. Some languages may use a different naming convention for this type of loop.

Example:-

```
#include<stdio.h>
int main(){
int i=1;
do{
printf("%d \n",i);
i++;
}while(i<=10);
return 0;
}
```




6.Comments:-Comments are the lines of non-executable code. They are one of the main aspects of all programming languages. The purpose of this is to provide information about the project, variable, or an operation. There are three types of comments in Dart programming:-

Make format comments: It is a single line comment (//) Example:- //Hello World.

Block Comments: It is a multi-line comment (/*...*/).Example:- /*Hello

World !!*/

Doc Comments: It is a document comment that used for member and types (///) Example:- /// Input Methods ///

7. Continue and Break:-

## Continue:-

Dart has also used the continue and break keyword in the loop, and elsewhere it required.

The continue statement allows you to skip the remaining code inside the loop and immediately jump to the next iteration of the loop.

We can understand it from the following example:-

### Example

```
void main() {
  for(int i=1;i<=10;i++){
    if(i==5){
      print("Hello");
      continue; //it will skip the rest statement
    }
    print(i);
  }
}
```

## Break:-

The break statement allows you to terminate or stop the current flow of a program and continue execution after the body of the loop.

The following example gives a detailed explanation:-

### Example

```
void main() {
  for(int i=1;i<=10;i++){
    if(i==5){
      print("Hello");
      break;//it will terminate the rest statement
    }
    print(i);
```

```
 }
}
```

## 8.Final and Const Keyword:-

### Final:-

The final keyword is used to hardcode the values of the variable and it cannot be altered in future, neither any kind of operations performed on these variables can alter its value (state).

We can use a final keyword to restrict the user. It can be applied in many contexts, such as variables, classes, and methods.

Const keyword is used to declare constant. We cannot change the value of the const keyword after assigning it.

### Example

```
void main() {
  final a = 100;
  const pi = 3.14;
  print(a);
  print(pi);
}
```

### Const:-

The Const keyword in Dart behaves exactly like the final keyword. The only difference between final and const is that the const makes the variable constant from compile-time only.

Using const on an object, makes the object's entire deep state strictly fixed at compile-time and that the object with this state will be considered frozen and completely immutable.

Example:-

```
void main() {
const ws1= "Ws Cube Tech";
print(ws1);
const String ws2= "Ws Cube Tech Again!!";
```

print(ws2);}

# Object Oriented Programming:-

1.Class:- A class is a collection of objects. It means the objects are created with the help of classes because every object needs a blueprint based on which you can create an individual object. A class definition includes the following things:-

Fields

Methods

Constructor

Getters and setters.

## 2. Object:-


An object is an entity, which has state and behaviour.

It can be physical or logical. In Dart, every value is an object, even primitive values like text and number.

Dart can also allow you to build your custom object to express more complex relations between data.

Example:-class person{

char name[20];

int id;

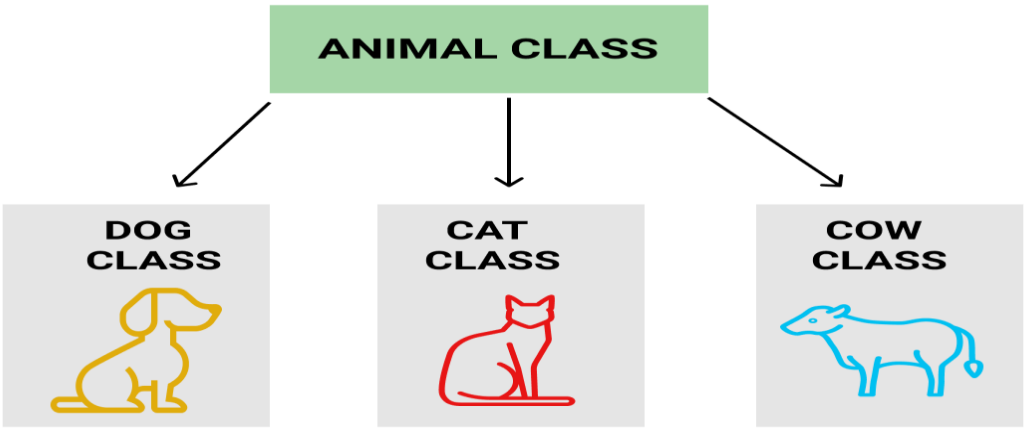
Public:

void getdetails(){}

};int main()

{person p1; // p1 is a object}

3. Inheritance:-




When one object acquires all the properties and behaviours of a parent object, it is known as inheritance.

It provides code reusability. It is used to achieve runtime polymorphism.

Inheritance is one of the most important features of Object-Oriented Programming.

a.Sub Class:- The class that inherits properties from another class is called Subclass or Derived Class.

b.Super Class:- The class whose properties are inherited by sub class is called Base Class or Super class.

c.Reusability:- Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

4.Polymorphism:-The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.


If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

a.Operator Overloading:-The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.

b.Function Overloading:- Function overloading is using a single function name to perform different types of tasks.Polymorphism is extensively used in implementing inheritance.

5.Abstraction:-

Data abstraction is one of the most essential and important features of object-oriented programming. Abstraction means displaying only essential information and hiding the details.

Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Hiding internal details and showing functionality is known as abstraction. For example phone calls, we don't know the internal processing.

Example:-

// Understanding Abstract class in Dart

// Creating Abstract Class

```
abstract class Ws {
        // Creating Abstract Methods
        void say();
        void write();
}

class WsCubeTech extends wscube{
        @override
        void say()
        {
                print("Yo Ws!!");
        }

        @override
        void write()
        {
                print("Ws Cube Tech");
        }
}main()
{
WsCubeTech ws1= new WsCubeTech ();
        ws1.say();
        ws1.write();}
```

Capsule

6.Encapsulation:-

In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Example:-

```
class Employee {
  // Private properties
  int? _id;
  String? _name;
// Getter method to access private property _id
  int getId() {
    return _id!;
  }
// Getter method to access private property _name
  String getName() {
    return _name!;
  }
// Setter method to update private property _id
  void setId(int id) {
    this._id = id;
  }
// Setter method to update private property _name
  void setName(String name) {
    this._name = name;
  }
}
```

```
void main() {
  // Create an object of Employee class
  Employee emp = new Employee();
  // setting values to the object using setter
  emp.setId(1);
  emp.setName("John");
  // Retrieve the values of the object using getter
  print("Id: ${emp.getId()}");
  print("Name: ${emp.getName()}");
}
```

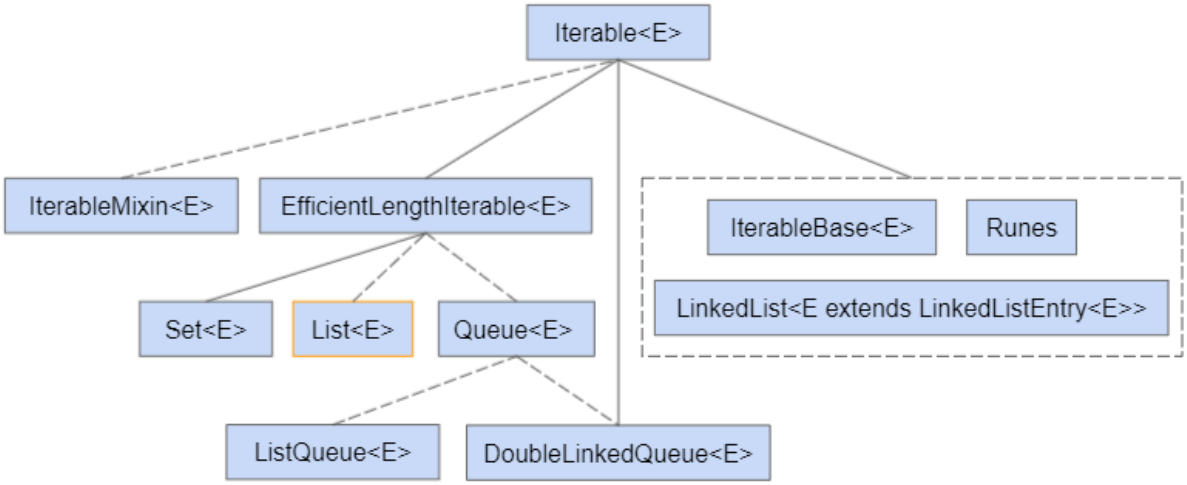
## 1.List <> :-

In most programming languages, an array is an indispensable concept. However, the Dart programming language does not have the traditional array concept. Instead, it uses List with similar features and adds new ones.

Below are the characteristics of List:-

Like traditional arrays, List contains ordered elements and is indexed starting at 0.

Depending on the type of List created, its size is fixed or can be automatically increased if necessary.




Example:-void main() {

```
  var lst = new List(3);
  lst[0] = 12;
  lst[1] = 13;
  lst[2] = 11;
  print(lst);
}
```

## 2.Map<K,V>:-

In the Dart language, the Map<K,V> class represents a data structure consisting of mappings between keys and values. Keys cannot be duplicated, and each key will correspond to a value.

One of the typical examples that can be mentioned here is a phone book. The phone number is the key and the owner name of the phone number is a value.

Example:-

```
void main() {
  var details = {'Usrname':'tom','Password':'pass@123'};
  print(details);
}
```

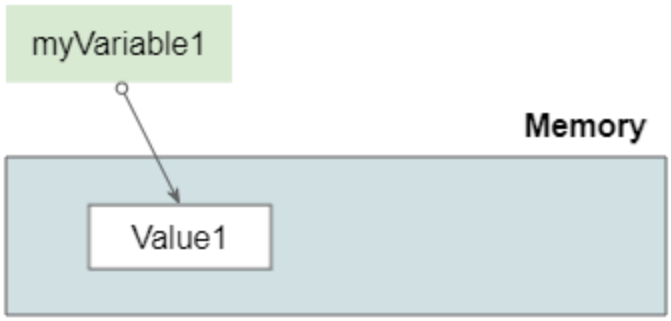
## 3.Variable:-

In programming, a variable is a value that can change, depending on conditions or on information passed to the program. Typically, a program consists of instructions that tell the computer what to do and data that the program uses when it is running.

In the Dart programming language, a variable is a name (an identifier) that points to an address where data is stored in memory.

Example:-

```
int main() {
  int a;
  int b;
}
```




## 4.Dynamic:-

If you create a variable using a dynamic keyword the data type assign automatically based on what variable you declared. For example, if your value is a string the dynamic keyword automatically detects the string data type.

The dynamic keyword is also used to declare a variable, which is quite similar to the var keyword. The difference is that the data type of this variable can be changed.

For Example:-

```
dynamic message = 'Hello, World';

message = 8;
```

# 5.Conditional Statements:-

In Dart, if-else statement is used to execute one of the two blocks: if-block or else-block; based on the result of a given condition.

In this tutorial, we will learn the syntax and usage of Dart If-Else statements.

## 1.Dart If-Else:-

Dart If-Else statement contains two blocks: if-block and else-block.

If the boolean_expression next to if keyword evaluates to true, then the code inside if-block is executed.

If the boolean_expression next to if keyword evaluates to false, then the code inside else-block is executed.

The syntax of if-else statements in Dart is given in the following.

```
if (boolean_expression) {
      //if block statement(s)
  } else {
      //else block statement(s)
  }
```

## 6.Final:-

The final keyword in Dart is used to create constants or objects that are immutable in nature. The only difference between the final and const keyword is that final is a runtime-constant, which in turn means that its

value can be assigned at runtime instead of the compile-time that we had for the const keyword.

Example:-

```
void main(){
  final int xy = 10;
   print(xy);
}
```

## 7.Creating First Program in Dart:-

```
import 'dart:io';
void main(){
    //read number from user
    print('Enter x');
    var x = int.parse(stdin.readLineSync()!);
    print('Enter y');
    var y = int.parse(stdin.readLineSync()!);
    var output = x + y;
    print('x + y = $output');
}
```

## 7.Const:-

A variable that is declared using the const keyword cannot be assigned any other value. Also, the variable is known as a compile-time constant, which in turn means that its value must be declared while compiling the program.

Example:-

```
void main(){
```

```
const name = "mukul";
  print(name);
  const marsGravity = 3.721;
  print(marsGravity);
}
```