# Lab 04

Instructor: Sidra Khatoon E-mail: skhatoon[@uit.edu](mailto:ad@uit.edu)

# Objective

The purpose of this lab session is to practice Flutter tool. Create a simple application on flutter.

**Student Information**

|  |  |
| --- | --- |
| **Student Name** |  |
| **Student ID** |  |
| **Date** |  |

**Assessment**

|  |  |
| --- | --- |
| **Marks Obtained** |  |
| **Remarks** |  |
| **Signature** |  |

# Objective

The purpose of this lab session is to practice Dart language.

# Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

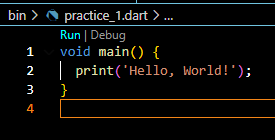
# How to Submit

Submit lab work using Teams.

**DART Language:**

**Hello World:**

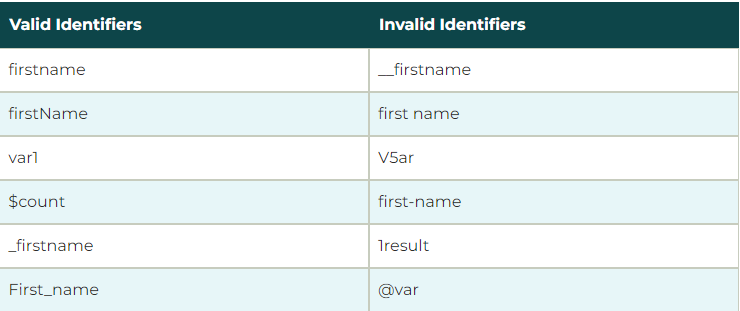
Every app requires the top-level main() function, where execution starts. Functions that don't explicitly return a value have the void return type. To display text on the console, you can use the top-level print() function:



**Dart Identifiers:**

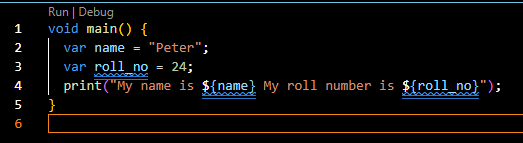
Identifiers are the name which is used to define variables, methods, class, and function, etc. An Identifier is a sequence of the letters ([A to Z], [a to z]), digits ([0-9]) and underscore(\_), but remember that the first character should not be a numeric.

* The first character should not be a digit.
* Special characters are not allowed except underscore (\_) or a dollar sign ($).
* Two successive underscores (\_\_) are not allowed.
* The first character must be alphabet (uppercase or lowercase) or underscore.
* Identifiers must be unique and cannot contain whitespace.
* They are case sensitive. The variable name Joseph and joseph will be treated differently.



**Dart Printing and String Interpolation**

The print() function is used to print output on the console, and $expression is used for the string interpolation. Below is an example.



**Semicolon in Dart**

The semicolon is used to terminate the statement that means, it indicates the statement is ended here. It is mandatory that each statement should be terminated with a semicolon (;). We can write multiple statements in a single line by using a semicolon as a delimiter. The compiler will generate an error if it is not use properly.

**Dart Whitespace and Line Breaks**

The Dart compiler ignores whitespaces. It is used to specify space, tabs, and newline characters in our program. It separates one part of any statement from another part of the statement. We can also use space and tabs in our program to define indentation

**Block in Dart**

The block is the collection of the statement enclosed in the curly braces. In Dart, we use curly braces to group all of the statements in the block. Consider the following syntax. And provide the proper format for the program. It makes code easy to understand and readable.

**Dart Datatypes**

The data types are the most important fundamental features of programing language. In Dart, the data type of the variable is defined by its value. The variables are used to store values and reserve the memory location. The data-type specifies what type of value will be stored by the variable. Each variable has its data-type. The Dart is a static type of language, which means that the variables cannot modify

# Dart Number

The Darts Number is used to store the numeric values. The number can be two types - integer and double.

## Integer - Integer values represent the whole number or non-fractional values. An integer data type represents the 64-bit non-decimal numbers between -263 to 263. A variable can store an unsigned or signed integer value. The example is given below –

int marks = 80;

## Double - Double value represents the 64-bit of information (double-precision) for floating number or number with the large decimal points. The double keyword is used to declare the double type variable.

double pi = 3.14;

### **Dart Strings**

A string is the sequence of the character. If we store the data like - name, address, special character, etc. It is signified by using either single quotes or double quotes. A Dart string is a sequence of UTF-16 code units.

var msg = "Welcome to JavaTpoint";

**Dart Boolean**

The Boolean type represents the two values - true and false. The bool keyword uses to denote Boolean Type. The numeric values 1 and 0 cannot be used to represent the true or false value.

bool isValid = true;

**Dart Lists**

In Dart, The list is a collection of the ordered objects (value). The concept of list is similar to an array. An array is defined as a collection of the multiple elements in a single variable. The elements in the list are separated by the comma enclosed in the square bracket[]. The sample list is given below.

var list = [1,2,3]

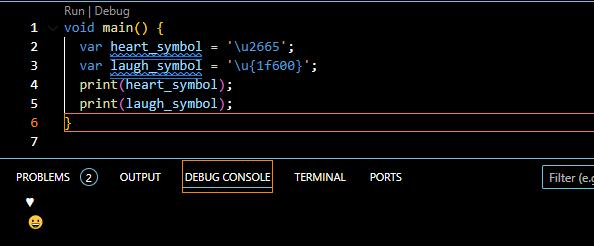
**Dart Maps**

The maps type is used to store values in key-value pairs. Each key is associated with its value. The key and value can be any type. In Map, the key must be unique, but a value can occur multiple times. The Map is defined by using curly braces ({}), and comma separates each pair.

var student = {'name': 'Joseph', 'age':25, 'Branch': 'Computer Science'}

**Dart Runes**

As we know that, the strings are the sequence of Unicode UTF-16 code units. Unicode is a technique which is used to describe a unique numeric value for each digit, letter, and symbol. Since Dart Runes are the special string of Unicode UTF-32 units. It is used to represent the special syntax.



### **Dart Symbols**

The Dart Symbols are the objects which are used to refer an operator or identifier that declare in a Dart program. It is commonly used in APIs that refers to identifiers by name because an identifier name can changes but not identifier symbols.

### **Dart Dynamic Type**

Dart is an optionally typed language. If the variable type is not specified explicitly, then the variable type is dynamic. The dynamic keyword is used for type annotation explicitly.

# Dart Variable

Variable is used to store the value and refer the memory location in computer memory. When we create a variable, the Dart compiler allocates some space in memory. The size of the memory block of memory is depended upon the type of variable. To create a variable, we should follow certain rules. Here is an example of a creating variable and assigning value to it.

var name = 'Devansh';

Here the variable called **name** that holds 'Devansh' string value. In [Dart](https://www.javatpoint.com/dart-programming), the variables store references. The above variable stores reference to a String with a value of Devansh.

## Rule to Create Variable

Creating a variable with a proper name is an essential task in any programming language. The Dart has some rules to define a variable. These rules are given below.

* The variable cannot contain special characters such as whitespace, mathematical symbol, runes, Unicode character, and keywords.
* The first character of the variable should be an alphabet([A to Z],[a to z]). Digits are not allowed as the first character.
* Variables are case sensitive. For example, - variable age and AGE are treated differently.
* The special character such as #, @, ^, &, \* are not allowed expect the underscore(\_) and the dollar sign($).
* The variable name should be retable to the program and readable.

## How to Declare Variable in Dart?

We need to declare a variable before using it in a program. In Dart, The **var** keyword is used to declare a variable. The Dart compiler automatically knows the type of data based on the assigned to the variable because Dart is an infer type language. The syntax is given below.

var <variable\_name>  = <value>;

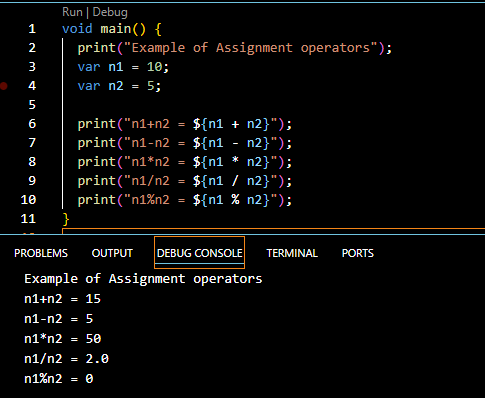
or

var <variable\_name>;

**Operators**

Dart supports the operators shown in the following table. The table shows Dart's operator associativity and [operator precedence](https://dart.dev/language/operators#operator-precedence-example) from highest to lowest, which are an **approximation** of Dart's operator relationships. You can implement many of these [operators as class members](https://dart.dev/language/methods#operators).

**Arithmetic Operators**

****

Increment and Decrement

++ and -- operators are known as increment and decrement operators and also known as unary operators, respectively. Unary operators, operate on single operand where ++ adds 1 to operands and -- subtract 1 to operand respectively. The unary operators can be used in two ways – postfix and prefix. If ++ is used as a postfix(like x++), it returns the value

of operand first then increments the value of x. If – is used as a prefix(like ++x), it increases the value of x.

Assignment Operator

= ,+= , -= , \*= , ~/= , %=

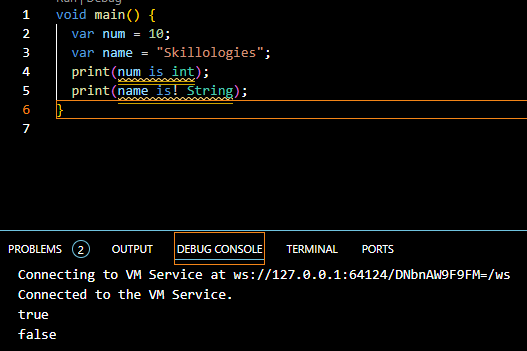
**Relational Operators**

== , != , < , > , <= , >=  
**Bitwise Operators**

AND &, OR | ,EX-OR ^ , >> , << , ~

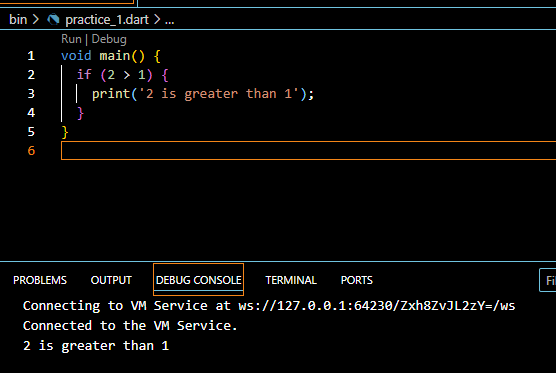
**Type Test Operators**

* as – It is used for typecast.
* is – It returns TRUE if the object has specified type.
* is! – It returns TRUE if the object has not specified type.

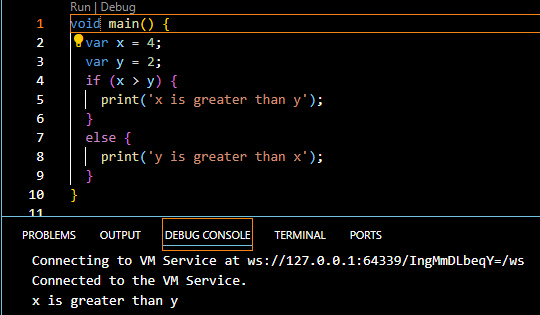


**Control Flow Statements**

**If statement:**



**Else-if statement:**



**If-else-if statement:**

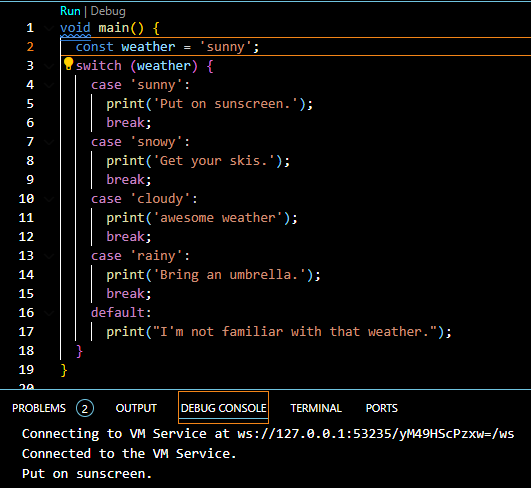
****

**What is wrong with this code?**

****

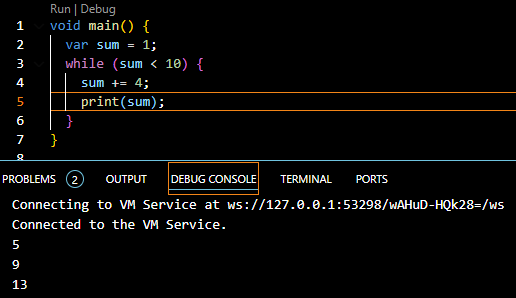
**Switch Statements**

An alternate way to handle control flow, especially for multiple conditions, is with a switch statement.



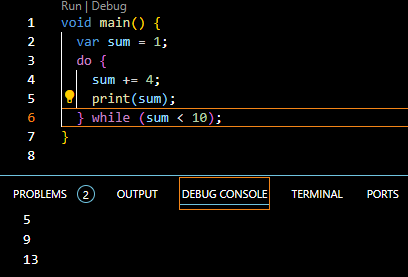
**While loop:**

A **while loop** repeats a block of code as long as a Boolean condition is true.



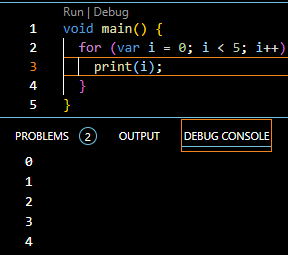
**DO-while loop:**

A variant of the while loop is called the do-while loop. It differs from the while loop in that the condition is evaluated at the *end* of the loop rather than at the beginning. Thus, the body of a do-while loop is always executed at least once.



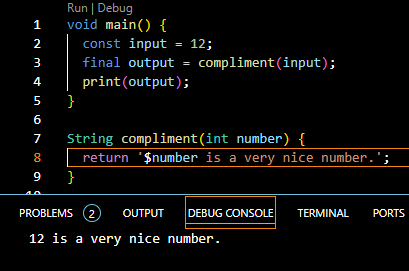
**For loop:**

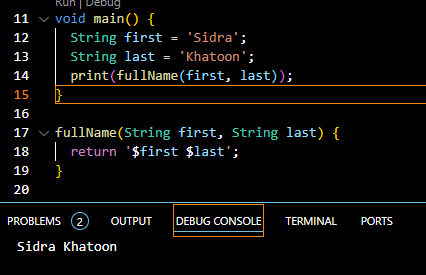
For loop is probably the most common loop you’ll see, and you use it to run a block of code a set number of times.

****

**Function:**

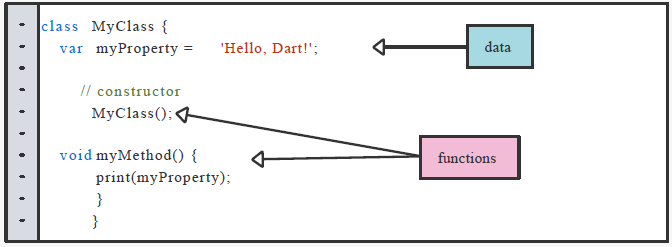
A **function** is one small task, or sometimes a collection of several related tasks, that you can use in conjunction with other functions to accomplish a larger task. In Dart, a function consists of a return type, a name, a parameter list in parentheses and a body enclosed in braces.



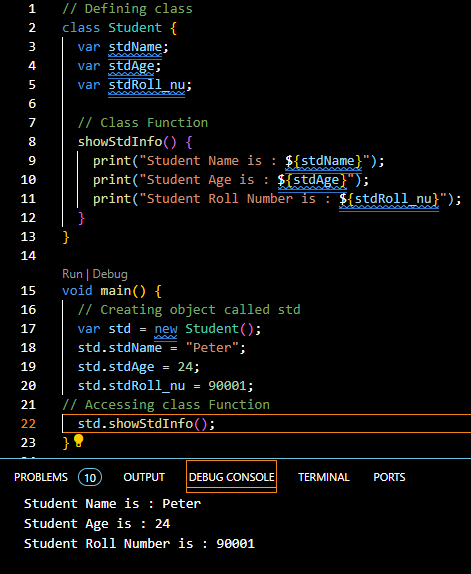
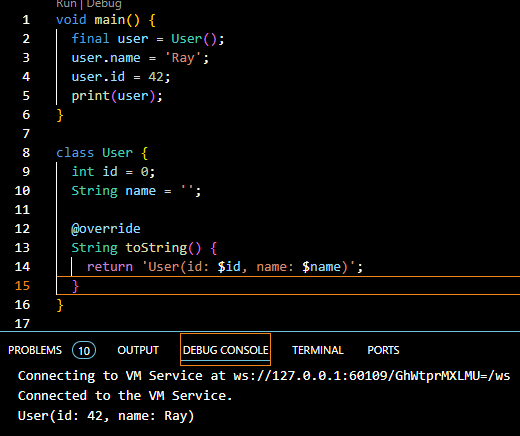


**Classes:**

Classes are a core component of object-oriented programming. They’re used to combine data and functions inside a single structure.



The functions exist to transform the data. Functions inside a class are known as methods, whereas constructors are special methods you use to create objects from the class.



**Encapsulation:**

Encapsulation is one of the important concepts of object-oriented programming. In Dart, Encapsulation means hiding data within a library, preventing it from outside factors. It helps you control your program and prevent it from becoming too complicated.

Encapsulation can be achieved by:

* Declaring the class properties as private by using underscore (\_).
* Providing public getter and setter methods to access and update the value of private property.

**Getter and Setter Methods:**

Getter and setter methods are used to access and update the value of private property. Getter methods are used to access the value of private property. Setter methods are used to update the value of private property.

For example, we will create a class named Employee. The class will have two private properties \_id and \_name. We will also create two public methods getId() and getName() to access the private properties. We will also create two public methods setId() and setName() to update the private properties.

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);

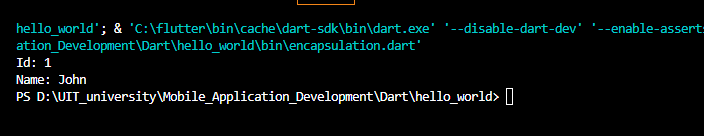
emp.setName("John");

// Retrieve the values of the object using getter

print("Id: ${emp.getId()}");

print("Name: ${emp.getName()}");

}



**Assessments**

1. Create a constant named myAge and initialize it with your age. Write an if statement to print out “Teenager” if your age is between 13 and 19 , and “Not a teenager” if your age is not between 13 and 19.
2. Write a function named youAreWonderful , with a string parameter called name . It should return a string using name , and say something like “You’re wonderful, Bob.”
3. Add another int parameter to above function called numberPeople so that the function returns something like “You’re wonderful, Bob. 10 people think so.”