Dart:

Dart is an object-oriented programming language, so it supports the concept of class, object. Every time a variable is created, much like in other languages, it has its own data type. In Dart language, there are the types of values that can be represented.

1. **Number:**
   It is used to hold the numeric value.

   - **int**: represents whole number.
   - **double**: represent decimal numbers
   - **num**: type is an inherited data type of the int and double types.

2. **String:**
3. It is used as a character sequence representation.
4. **Boolean:**
   It represents Boolean values true and false In DART, a Boolean literal is represented by the keyword **bool**.
   Lists:
   The list data type in computer languages has similarities to arrays. A list is used to represent a collection of objects.
5. **Maps:**
   A combination of a key and its value pair form the Map object. Keys and values on a map may be of any type. It is a dynamic collection.

Conditional statement:

Statements that allow the programmers to decide which statement should run in different conditions.

1. **If else statement:**

Types:

- **If Statement:**
  This type of statement acts like a condition check; if the condition is true, the statements inside it are executed .If not, the statements are neglected by the code.

  Syntax:

```
if (condition ){
        // statements
}
```

- **If-else Statement:**
  This type of statement acts like a condition check; if the condition is true, the statements inside it are executed .If not, the else statement is executed.

Syntax:

```
if (condition ){
  // statements
}
else {
  // statements
}
```

- **else-if Ladder:**

  This kind of statement only verifies the condition; if it is true, the statements inside are executed if not, alternative if conditions are checked; if they are true, they are executed. if not, alternative if conditions are checked. This process is continued until the ladder is completed.

  Syntax:

```
if ( condition1 ){
  // statement
}
else if ( condition2 ){
  // statement
}
.
.
.
else {
  // statement
}
```

- **Nested if Statement:**

  This type of statement checks the condition and if it is true then the if statement inside it checks its condition and if it is true then the statements are executed otherwise else statement is executed. Sometimes you need to check a condition inside another condition.

  Syntax:

```
if ( condition1 ){
  if ( condition2 ){
      // statements
  }
  else {
  // statements
  }
}
```

- **Multiple if Statement:**

  This type of statement checks the condition and if it is true then the if statement inside it checks its condition and if it is true then the statements are executed else statement is not involved in

it. You can write multiple if statements one after another.Each condition is evaluated independently.

Syntax:

```
void main() {
  int number = 10;

if ( condition1 ){
      // statements
  }

  if ( condition2 ){
      // statements
  }

if ( condition3 ){
      // statements
  }
}
```

2. **Switch case statement:**

A Switch case is used to execute the code block based on the condition.

Syntax:

```
switch(expression) {
   case value1:
      // statements
      break;
   case value2:
      // statements
      break;
   case value3:
      // statements
      break;
   default:
     // default statements
}
```

- The expression is evaluated once and compared with each case value.
- If expression matches with case value1, the statements of case value1 are executed. Similarly, case value 2 will be executed if the expression matches case value2. If the expression matches the case value3, the statements of case value3 are executed.
- The break keywords tell dart to exit the switch statement because the statements in the case block are finished.
- If there is no match, default statements are executed.

What is the difference between switch case and if else?

- Switch is generally faster than a long list of ifs because the compiler can generate a jump table. The longer the list, the better a switch statement is over a series of if statements.
- In switch case, We can use equal & OR operator.

LOOP:

Loop is used to run a block of code repetitively for a given number of times or until matches the specified condition.

Types of Loop:

1. **Pre condition/ Pre level Loop:**

- **Do while Loop:**
  Do while loop is used to run a block of code multiple times. The loop's body will be executed first, and then the condition is tested. In Dart, do-while loop is used to execute a block of code atleast once and then repeat the execution as long as specified condition remains true. The condition is checked after the block of code has been executed.

Syntax:

```
do{
   statement1;
   Increment/decrement;
}while(condition);
```

2. **Post Conditional/ Post level Loop:**

- **For Loop:**

  It is used to run a code block multiple times according to the condition.

Syntax:

```
for (initialization; condition; increment/decrement){
                    statements;
            }
```

- **While Loop:**

The body of a while loop executes until and unless the condition is true. Conditions must be written before statement. This loop checks conditions on every iteration. The code enclosed in {} is run if the condition is true. If the condition is false, then the loop stops.

Syntax:

```
while(condition){
    //statement(s);
   // Increment (++) or Decrement (--) Operation;
}
```

What is terminating value?

A terminating value is often used to indicate when a loop should stop.

1. **Construct Even program in dart using loops:**

```
for (i=0; i<=10; i+2){
        print(i);
}
```

Output:

2,4,6,8,10

2. **Construct Odd program in dart using loops:**

```
for (i=0; i<=10; i+1){
   print(i);
 }
```

Output:

1,3,5,7,9

3. **Construct Factorial program in dart using loops:**

```
var num1=6;
var fact=1;

for (var i=1; i<=num1; i++){
        fact= fact*i;
        print(fact);
}
```

LIST:

- One of the most popular data structure in dart is List. Dart represents arrays in the form of List objects. A List is simply an ordered group of objects.
- Each element in the List is identified by a unique number called the **index**. The index starts from **zero.**The index is also referred to as the **subscript.**

Why list is used in dart?

List is used to representing a collection of objects.

Methods:

1. **Insert List item in Dart:**

- **add**:
  Add single value to the end of this list.
- **addall**:
  Add multiple value to the end of this list.
- **Insert**:
  Inserts element at position index in the list.
- **Insertall**:
  Inserts all objects at position index in this list.

What is the difference between addall and insertall?

Addall is for straightforward cases where you want to append items to the end of the list. Whereas insertall is when you need to place new items at a specific position within the list.

2. **Update List item in Dart:**

- **Update by Index:**
  Replace element by index number.
- **Update by replace range:**
  Replaces a range of elements with the elements of replacements.

3. **Remove items from List in Dart:**

- **remove:**
  Removes the first occurrence of value from this list.
- **removeAt:**
  Removes the object at position index from this list.
- **removeLast:**
  Removes and returns the last object in this list.
- **removeRange:**
  Removes a range of elements from the list.
- **removeWhere:**
  remove all the items that match a given condition

Prepared By: Hiba Naveed

4. **Other Methods:**

Type equation here.

- **Length:**

The number of objects in this list.

- **Reversed:**

Elements in this list in reverse order.

- **Last:**

The last element.

- **First:**

The first element.

- **isEmpty:**

Whether this collection has no elements.

- **isNotEmpty:**

Whether this collection has at least one element.

- **Clear:**

Removes all objects from this list; the length of the list becomes zero.

- **shuffle:**
  This method re-arranges order of the elements in the given list randomly.

KEYWORDS IN DART:

1. **Dynamic:**
   It is used to declare dynamic variable that can store any type of value in it, either string or int.
   Example:
   Dynamic std= ”Hiba”;
   Std=40;
- Dynamic keyword is more flexible and allows variable to hold any type of value at run time.

2. **Var:**
   It is used to declare variable that can store any type of value initially. However, the type is inferred from initial, it cannot be changed.
   Example:
   Var std= ”Hiba”;
   Std=40;
- Here 40 is of int type and Hiba is of string type. So, it will result as a type error.

What is the difference between dynamic and var?

Dynamic variable type can be changed at run time but var type cannot be changed at run time.

3. **Const:**
   Const is used for making variable constant throughout the program. Also known as compile time constant and it can't be changed.
   Example:
   Const pi= 3.142
   - Any attempt to modify const value will result in compile time error.

4. **Final:**

   Final values can only be set once but it is assigned at run time.
   Example:
   Final currentdate= DateTime.now();

OOP (Object Oriented Programming):

Object-Oriented Programming (OOP) is a programming based on the concept of "objects," which can contain data and methods.

Class:

Classes are blueprints for creating objects. It defines a type of object by bundling data (fields) and methods (functions) that operate on the data.

OOP principles include Encapsulation, Inheritance, Polymorphism, and Abstraction.

1. **Encapsulation:**
   It is used for data binding and data hiding.
2. **Inheritance:**
   Inheritance allows a class to inherit properties and methods from another class. The class that inherits is called a subclass/ Derived class/Child Class, and the class it inherits from is called a superclass/ Base class/ Parent class.
   - Extend is used to relate two classes.
3. **Polymorphism:**
   Polymorphism allows objects to be treated as instances of their parent class, even if they are instances of derived classes. It typically involves method overriding, where a subclass provides a specific implementation of a method that is already defined in its superclass.
   - **Overriding:**
     It allows a subclass to provide a specific implementation for a method that is already defined in its superclass.

- **Overloading:**
  It is a feature found in some object-oriented programming languages where multiple methods can have the same name but differ in their parameters (number, type, or both). It allows a class to have more than one method with the same name but different signatures, making it easier to perform similar operations with different inputs. Dart does not support overloading method.

4. **Abstraction:**
   Abstraction involves creating abstract classes and methods that can be inherited and implemented by subclasses. It hides complex implementation details and shows only the necessary features of an object.

Can we use overloading in dart?

Dart does not support method overloading in the traditional sense, as found in languages like Java or C++. In those languages, you can define multiple methods with the same name but different parameter lists (either in number, type, or order of parameters).

What is the difference between argument and parameter?

The values that are declared within a function when the function is called are known as an argument. The variables that are defined when the function is declared are known as parameters.