Ctrl + Shift + P → new project

Date 29/5/25
Page

VS code Dart Learning

→ files add in 'lib' folder
→ Extension .dart

① first program

→ don't return anything

```
void main() {
    print("Hello world");
}
```

To run tab on ▷ button

② Dart Syntax

for printing on output screen :

```
Stdout.write("  ");   → gets new line
print("  ");          → no new line
```

⇒ stdout.write(" ");

↓

Standard output

used for inputs

⇒ stdin.readlineSync()!

- stdin → standard input
- readlineSync() → reads particular line
  - ↳ by default takes string value

⇒ void main()

- ↳ Cannot return anything (nothing returns)
- ↳ main method for execution

(2) Addition program

```
void main() {
  int num1 = 30, num2 = 50;      //initialize variables
  int sum = num1 + num2;         //sum of two numbers
  print(sum);
}
```

Output:- 80

int.parse → to convert 'readLineSync' to int
ie [user input converts to int]

③ Addition → User input

```
import 'dart:io';
void main(){
   stdout.write("Enter first number:");
   int num1 = int.parse(stdin.readLineSync()!);

   stdout.write("Enter second Number:");
   int num2 = int.parse(stdin.readLineSync()!);

   int sum = num1+num2;
   print(sum);
}
```

output

```
Enter first number: 50
Enter second number: 50
100
```

Settings
↓
search Dart cli
↓
Dart Cli console
↓
Selects Terminal to enable writing in console

Dart Comments

↳ Non Executable statements

Two types
- Singleline
- multiple line

// → for single line
/* */ → for multiple line

Dart Datatypes

1. Int → used for simple integer numbers
2. String → used for set of letters or words.
3. double → used for Point values
4. Bool → has only two values → True / false

4. String Program

```
void main(){
    Print("Enter name");
    string name = stdin.readLineSync()!;
    Print(name);
}
```

Output → Enter name: Tisha
Tisha

Keywords

↳ Reserved or predefined words

→ 61 keywords in Dart

Ex:- final
const
double
string
int ....

Variables

↳ container that stores values

```
var Ram = 10;
var Shyam = 20;
var sum = Ram+Shyam;
print(sum);
```

↓ ↓ ↓

30

Operators

↳ special symbols used for specific purpose

Types:- Arithmetic operators
Ternary operator (conditional operator) → ? :
logical operator (&& ||)

(7) Ternary Operator

```
void main() {
  stdout.write("Enter num : ");
  int num = int.parse(stdin.readLineSync()!);

  num % 2 == 0 ? print("Even") : print("odd");
}
```

Constants

→ (const) ⟶ value cannot be change.

Ex:- const int a=12;

⇒ void main() {
    const int a=20;
    print(a);
}

Integers

↳ Datatype
↳ Numeric datatype.

Map → Stores in unordered way.

Date
Page

String → functions
.length → shows length
→ Print(name.length);

↳ inside double quotes
↳ in words

Ex:- String name = "Tisha";

Collections & Lists
↳ Static List (fixed values)
Dynamic List (values can be change)
↳ set of objects

⇒ List stores data in ordered way (just like array)

List commands :-
1. listName.add(num);
2. listName.remove(value);
3. listName.removeAt(0); → position index starts from 0
4. listName.contains(value); → returns boolean value
5. listName.length();
6. listName.isEmpty();
7. listName.isNotEmpty();
8. ListName.insert(1, 2);
   Ex:- list<int> listName = [1, 2, 3, 4, 5];
   ↓
   replace with 2.
9. listName.insertAll(0, [3,4,5,6,]);
   ↳ add list inside list)
10. listName.first(); → shows what is first num
11. listName.last(); → returns last value.

List → list<int> ListName = [ ]; → ordered way

Set → set setname = { }; → unique items, unordered way

Map → unordered way, duplicate value can be stored.

Sets

↳ stores in unordered way
↳ Only unique items can be stored

Ex:- Set set = {1, 2, 3, 4, 5};

(1)
```
class Test {
    Set add() {
        Set items = {1, 2, 3, 4, 5};
        print(items);
    }
}
void main(){
    Test obj = Test();
    obj.add();
}
```

→ Synchronous Programming

Date
Page

Commands in set

① items.add(6); ⟶ ~~inse~~ add at last position, value
② items.isEmpty();
③ items.isNotEmpty();
④ items.last(); → for getting last value.
⑤ items.first(); → " " first "
⑥ items.length(); → returns length
⑦ items.contains(6); → if 6 value is there → it returns True
↳ else returns false.
⑧ items.elementAt(0); → returns 1 because → 0th index position value
⑨ items.remove(3); → removes value 3.
⑩ items.addAll(item1); → adds 2 items.

Ex:- set items = {1, 2, 3, 4, 5};
set items1 = {6, 7, 8, 9};
items.addAll(item1);
print(items); ⟶ because we added(items1) in 'items' set.

MAP

↳ stores value in unordered way
↳ stores in key (→ unique), value
("id" : "value")

→ map is used when we want to store multiple values in one variable.

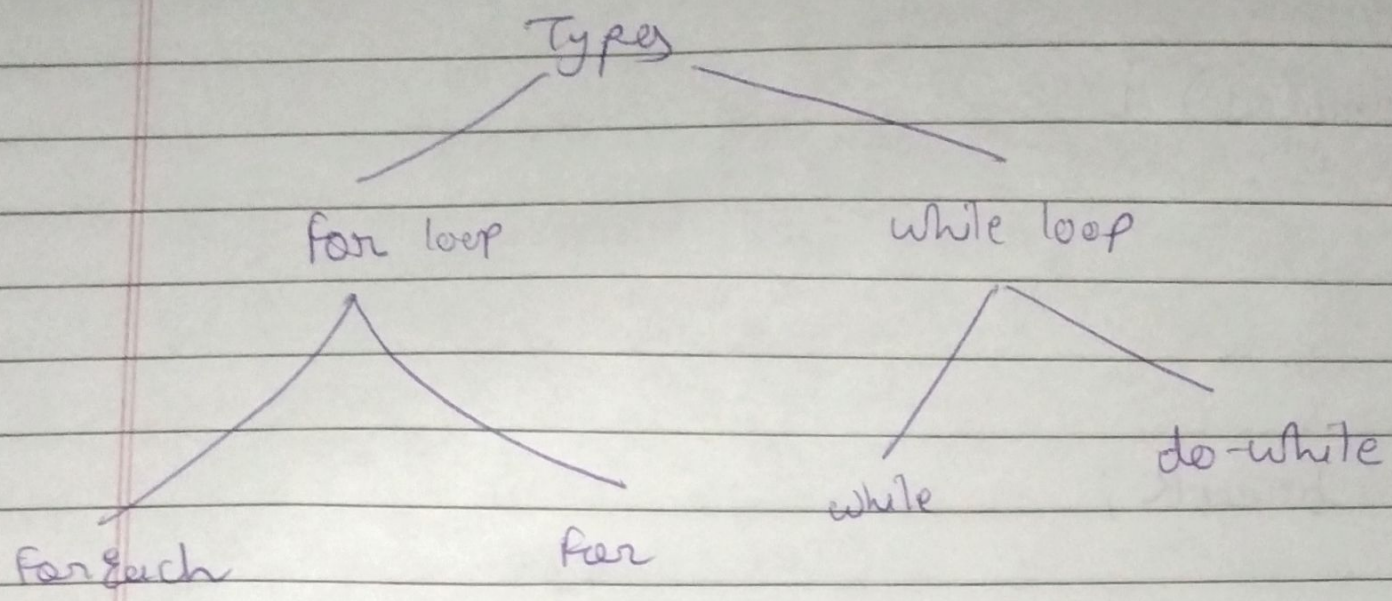
code

```
class Test{
    add(){
        Map<String, String> mapdata = {
            "Name" : "Tisha",
            "Code" : "Dart"
        };
        print(mapdata);
    }
}
void main(){
    Test obj = Test();
    obj.add();
}
```

key datatype
value datatype
map name

Loops

↳ Redundancy less.



for loop

↳ when we know Starting & Ending

```
for (starting point , Ending point , Increment/Decrement) {
        ↓                    ↓
    initialization       condition

}
```

while

↳ when we don't know starting point.

```
while (condition) {
    //code
    ++, --
}
```

Date
Page

Do while

→ when we don't knew about starting point, Ending point.

```
do {
    // code
    ++ , --
} while (Ending point)
```