Instance Variables: These are variables declared inside a class but outside any method, and belong to the instance of the class.

Local Variables: Local variables are variables that are declared inside a method, constructor, or block, and can only be used within that scope.

Ex:
```
class Sample
  {
     String? name;  //Instance variable
     int? age;
     Sample(String? name,int? age)    //local variable
     {
       this.name=name;
       this.age=age;
     }
     void display()
     {
        print("${name} and ${age}");
     }

  }
void main()
{
  Sample s=Sample("john",22);
  s.display(); // john and 22
}
```

Null Assertion Operator: In Dart, the null assertion operator (!) is used to forcefully convert a value of a nullable type to a non-nullable type , by asserting that the value is not null at that specific point in the code.

Ex: class Sample

```
    {
      String? name;
      Sample(this.name);
      void dis()
      {
        if(name!=null)
        {
          print("${name!.length}");
        }
        else
        {
          print("name is null");
        }
      }
   }
void main()
{
  Sample s=Sample(null);
  s.dis(); //name is null
}
```

Null Aware Access Operator: In Dart, the null-aware access operator `?.` is used to safely access a member (property or method) of an object only if the object is not null. If the object is null, the expression returns null instead of throwing an error.

Ex: class Sample

```
    {
```

```
    String? name;
    Sample(this.name);
    void dis()
    {
      print("${name?.length}");
    }
  }
void main()
{
 Sample s=Sample(null);
 s.dis(); //null(it won't throw any error)
}
```

Constructors: Constructor has same name as class name and it is automatically called and used to initialize values of the objects.

Named Constructors: It allows you to create multiple constructor inside same class, each with different name.

Ex:
```
class Sample
  {
    String? name;
    int? age;
   Sample(this.name,this.age)
   {
     print("${name} and ${age}");
   }
  Sample.call(String? msg)
```

```
    {
        print("Welcome to ${msg}");
     }
  }
void main()
{
  Sample s=Sample("john",22);  // john and 22
  Sample s1=Sample.call("i-exceed");   //Welcome to i-exceed
}
```

Late Instance Variable: It is used to delay the initialization of a variable until its actually used, but gurantee that is will be non-null.

Ex:
```
class Sample
    {
      late String? name;
      Sample(this.name)
     {
         print("$name");
      }
   }
 void main()
{
  Sample s = Sample(null);   //null
}
```

This keyword: this refers to the current instance of the class. It is used to access the instance variables and methods of the object within the class, especially when there is a naming conflict between instance variables and parameters.

1) Differentiate between instance variables and local variables:

Ex: class Sample
```
    {
       String? name;  //Instance variable
       int? age;
       Sample(String? name,int? age)    //local variable
       {
         this.name=name;
         this.age=age;
       }
       void display()
       {
         print("${name} and ${age}");
       }

    }
void main()
{
    Sample s=Sample("john",22);
    s.display(); // john and 22
}
```

2) Construtor Chaining:

Ex: class Sample
```
    {
     int val=0;
     Sample inc()
     {
       val++;
       return this;
     }
     Sample dec()
     {
       val--;
       return this;
     }
   }
void main()
{
  Sample s =Sample().inc().dec().inc();
  print(s.val); //1
}
```

3) Named Constructor(this):

Ex: class Customer

```
{
 String name;
 int id;
 String branch;
 Customer(this.name, this.id, this.branch);
 Customer.onlybranch(String branchname) : this("", 0, branchname);
 void display()
 {
   print('Name: $name, ID: $id, Branch: $branch');
 }
}

void main()
{
  Customer c = Customer.onlybranch("Downtown");
  c.display();  //  Name: , ID: 0, Branch: Downtown
}
```

Factory Constructor: A factory constructor in Dart is a special type of constructor that does not always create a new instance of the class. Instead, it can:

- Return an existing instance,
- Return a subclass instance,
- Perform additional logic before returning an object.

Ex: class Sample

```
{
   String? name;
   String? pass;
   Sample._(this.name,this.pass);
   factory Sample.callme(String name,String pass)
   {
     return Sample._(name,pass);
   }
   void dis()
   {
     print("${name}:${pass}");
   }
}
void main()
```

```
{
  Sample s=Sample.callme("john","123john");
  s.dis(); //john:123john
}
```