CS 315

Homework Assignment 3

Subprograms in Dart

Zeynep Selcen Öztunç

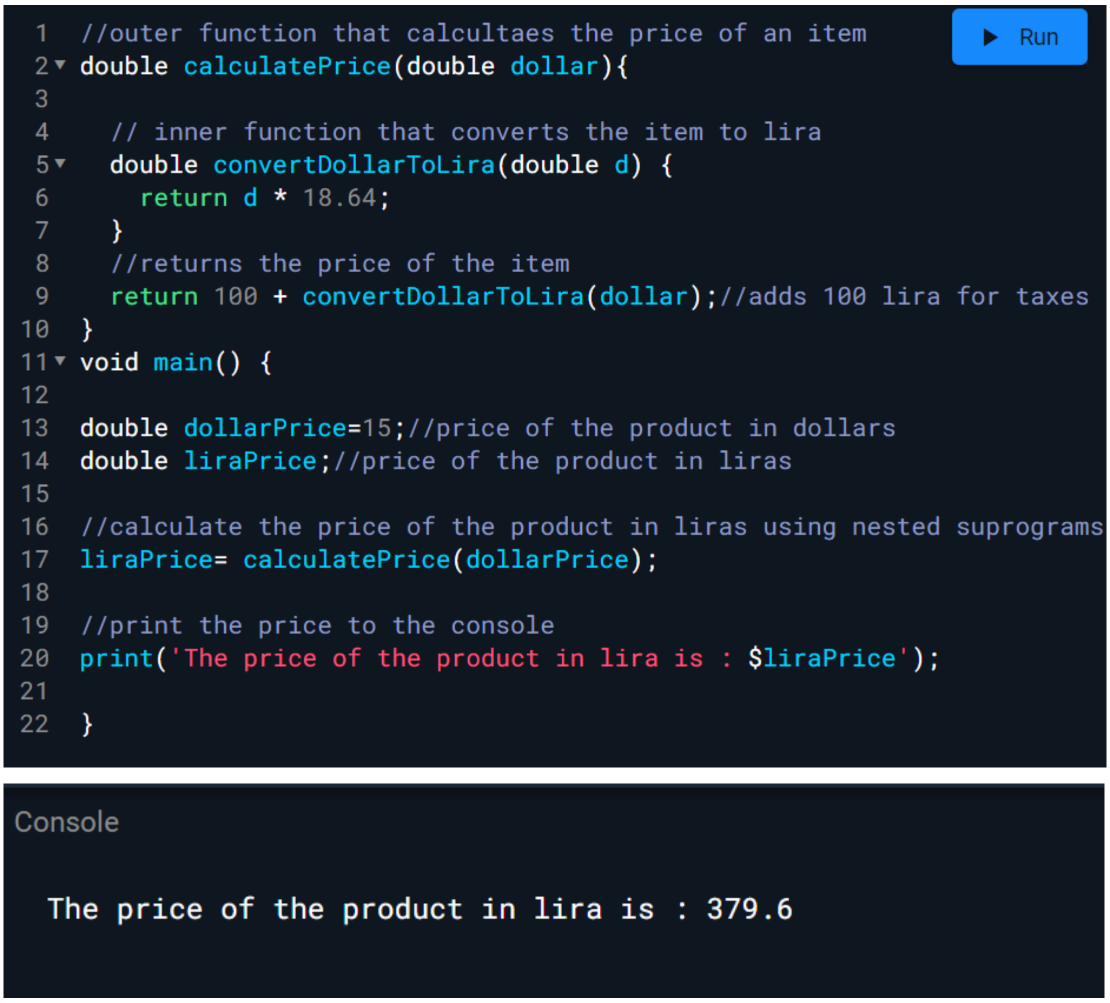
21902941

Table of Contents

1.Nested subprogram definitions

Dart language allows nested subprograms since it is a truly object oriented language. The inner function's variables and arguments cannot be used by the outer function. However, the inner function can access the variables and arguments of the outer function since it is inside the outer function's scope [1].

Example code below depicts the use of nested subprograms in Dart. In the code, there are two nested functions called calculatePrice and convert dollarToLira. Using its parameter *dollar*, the outer function calls the inner function convertDollarToLira in its return statement. Since the inner function has access to the variables of the outer function, it uses the outer function's parameter *dollar* to convert the price and do the necessary calculations. In the main function, these nested functions are tested.

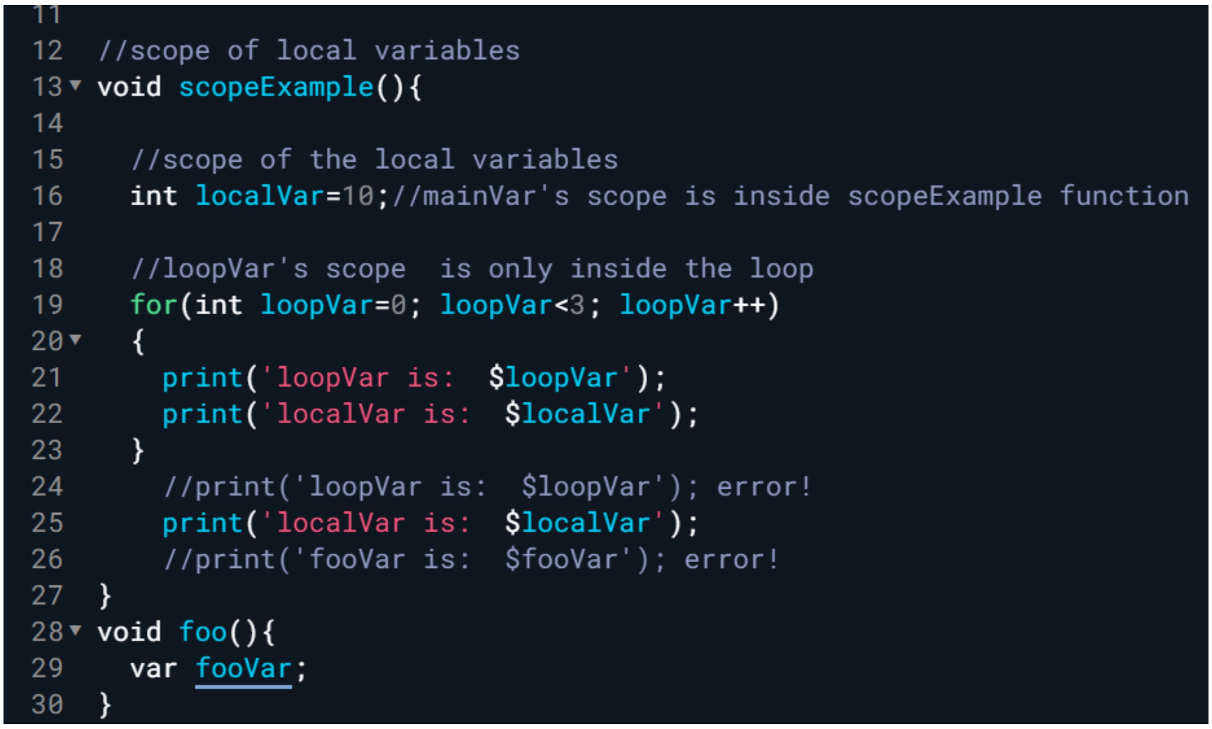


```
//outer function that calcultaes the price of an item
double calculatePrice(double dollar){

  // inner function that converts the item to lira
  double convertDollarToLira(double d) {
    return d * 18.64;
  }
  //returns the price of the item
  return 100 + convertDollarToLira(dollar);//adds 100 lira for taxes
}
void main() {

double dollarPrice=15;//price of the product in dollars
double liraPrice;//price of the product in liras

//calculate the price of the product in liras using nested suprograms
liraPrice= calculatePrice(dollarPrice);

//print the price to the console
print('The price of the product in lira is : $liraPrice');

}
```

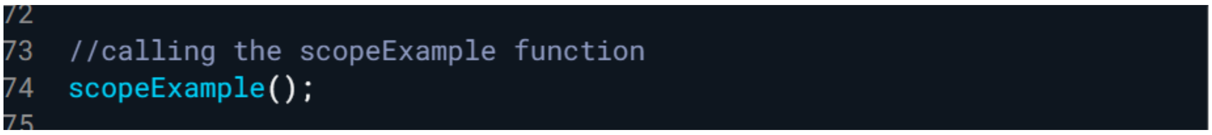
```
The price of the product in lira is : 379.6
```

2.Scope of local variables

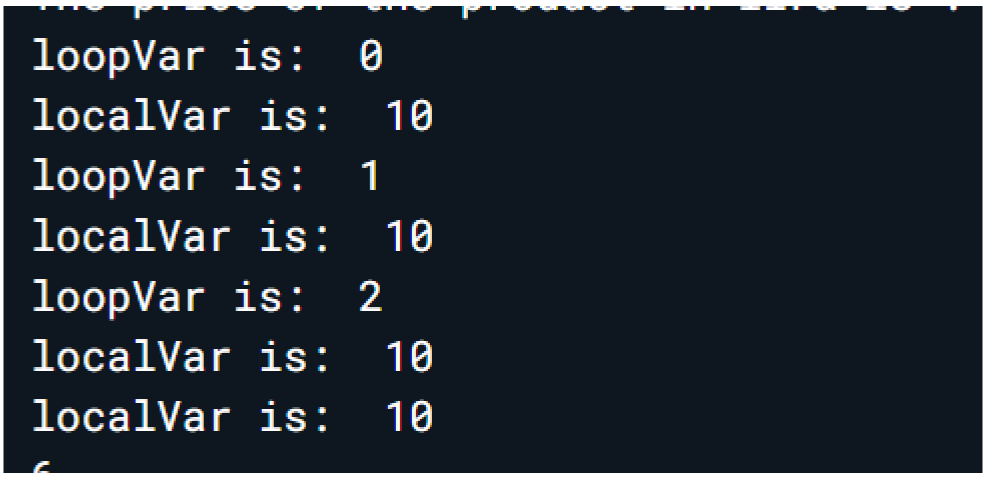
In the Dart language, the scope of local variables is only inside of the block or function that they are defined in. Once the execution of the block or the function has ended, it can be said that the scope of the function has ended[2]. In the example code below, localVar is defined in the function scopeExample(), that means it can be accessed from everywhere in the scopeExample() function. However, loopVar is only defined in the for loop, so it is only visible inside the for loop. Trying to print it outside of the for loop gives an error. Variable localVar can be printed inside the loop, since the loop is inside the function. Also trying to print the variable fooVar in the function scopeExample() also gives an error, since the fooVar's scope is the function where it is defined which is foo(); so it cannot be accessed outside of foo().



```
//scope of local variables
void scopeExample(){

  //scope of the local variables
  int localVar=10;//mainVar's scope is inside scopeExample function

  //loopVar's scope  is only inside the loop
  for(int loopVar=0; loopVar<3; loopVar++)
  {
    print('loopVar is:  $loopVar');
    print('localVar is:  $localVar');
  }
    //print('loopVar is:  $loopVar'); error!
    print('localVar is:  $localVar');
    //print('fooVar is:  $fooVar'); error!
}
void foo(){
  var fooVar;
}
```



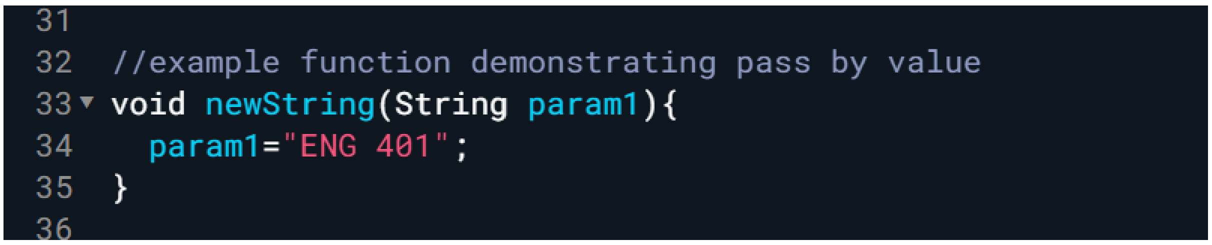
```
//calling the scopeExample function
scopeExample();
```



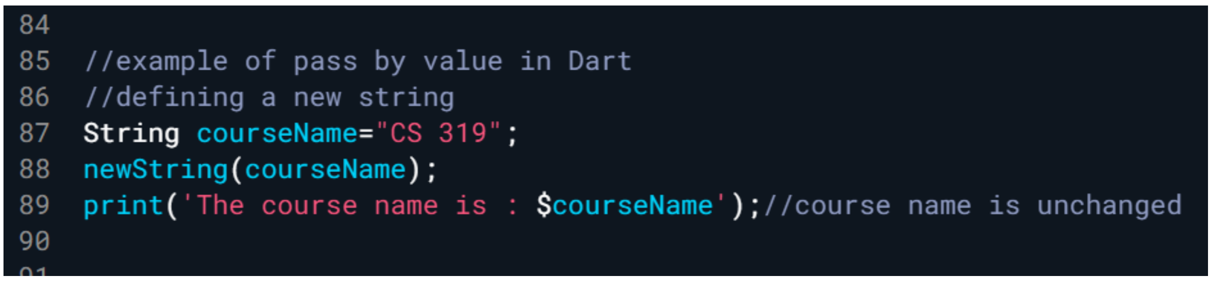
```
loopVar is:  0
localVar is:  10
loopVar is:  1
localVar is:  10
loopVar is:  2
localVar is:  10
localVar is:  10
```

3.Parameter passing methods

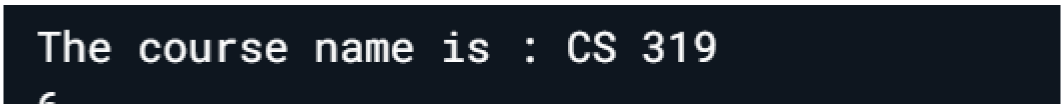
Dart language only supports pass by value, it does not support pass by reference. In the pass by reference, the method parameter values are copied to another variable, and the copied object is passed to the function. So the function does not use the actual object but it rather uses the copy of the object [3]. In the example code below, the value of the string courseName is set to “CS 319”. Then we pass this variable to the function newString() as an argument. Inside newString(), the value of its parameter param1 is set to “ENG 401”. However, when we try to print the value of courseName in the main after calling the function newString(), the value of it is still “CS 319”. This is due to the fact that the parameters are passed by value- so a copy of courseName is used inside the function, therefore even after the execution of the function, variable courseName is unchanged.



```
//example function demonstrating pass by value
void newString(String param1){
  param1="ENG 401";
}
```



```
//example of pass by value in Dart
//defining a new string
String courseName="CS 319";
newString(courseName);
print('The course name is : $courseName');//course name is unchanged
```

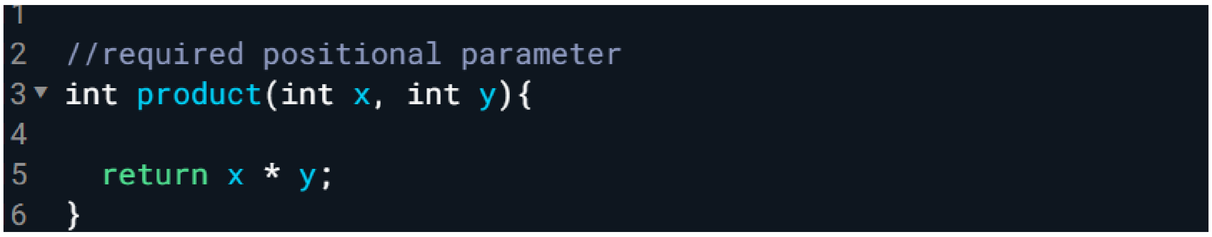


```
The course name is : CS 319
```

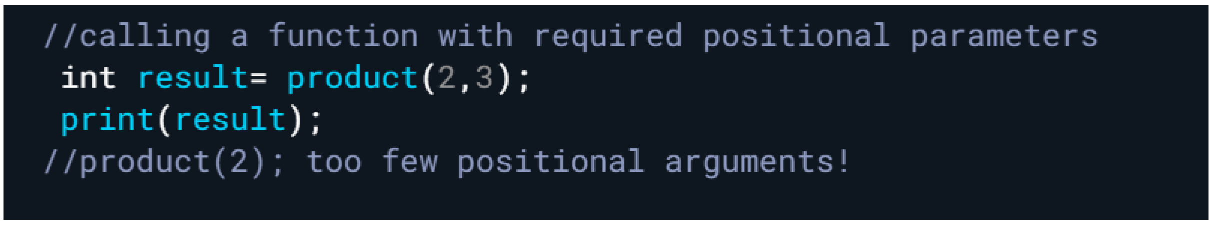
.

4.Keyword and default parameters

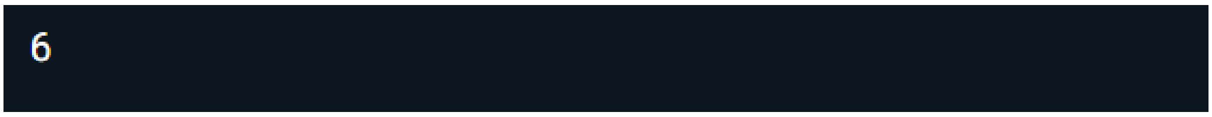
In the Dart Language, there are positional and named parameters. Positional parameters are parameters which are linked by their position. They must be specified in the order in which they appear [4]. They are declared with a type and a name. The example below shows a function with required positional parameters. When the function is called from main, the value for both of the parameters, x and y, should be given. The order in which they are called matters too. In my example below, x has the value 2 and y has the value 3. The function cannot be called with a single argument, since the positional parameters are required; both of the arguments should be given.



```
//required positional parameter
int product(int x, int y){

  return x * y;
}
```

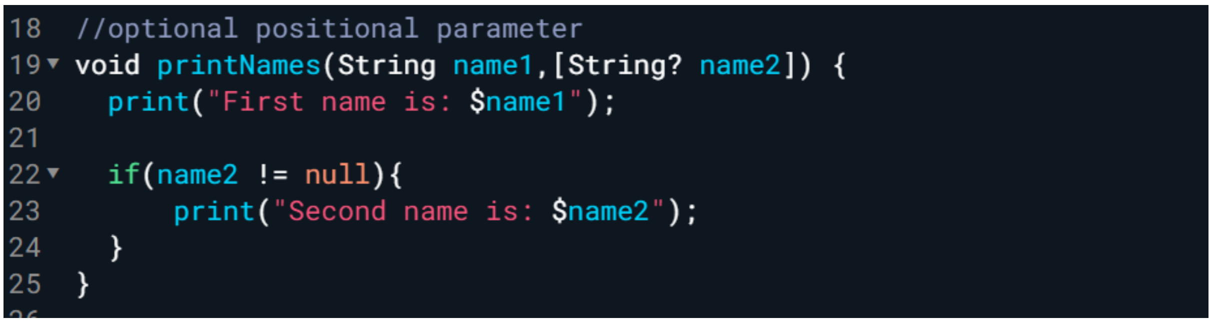


```
//calling a function with required positional parameters
 int result= product(2,3);
 print(result);
//product(2); too few positional arguments!
```

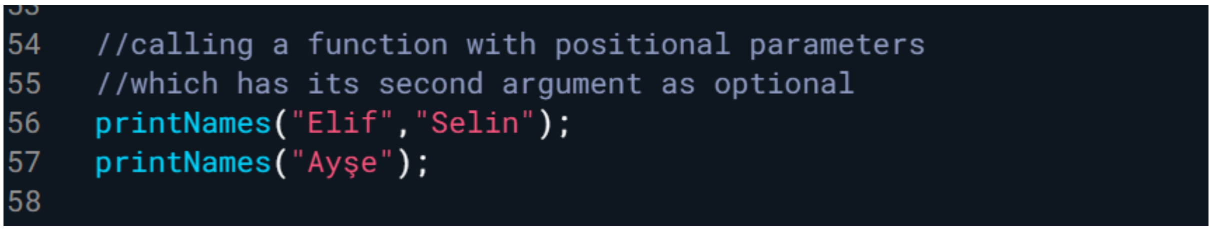


```
6
```

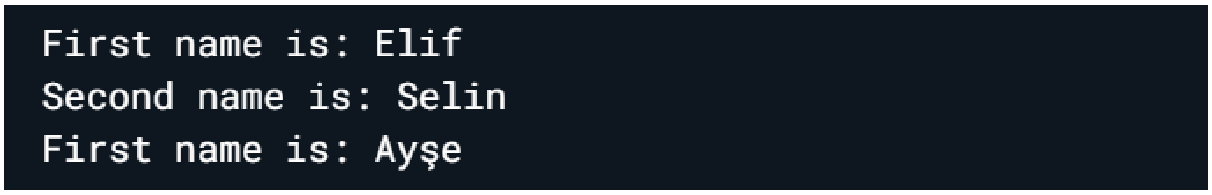
Positional parameters can also be optional, as depicted in the example below. To make a positional parameter optional, it should be stated inside “[]”. If the parameter is optional, it is not required to be given as an argument when it is called. As it can be seen from the example below, when the printNames function, which has its second parameter as an optional parameter, is called with a single argument, no error is given.



```
//optional positional parameter
void printNames(String name1,[String? name2]) {
  print("First name is: $name1");

  if(name2 != null){
      print("Second name is: $name2");
  }
}
```

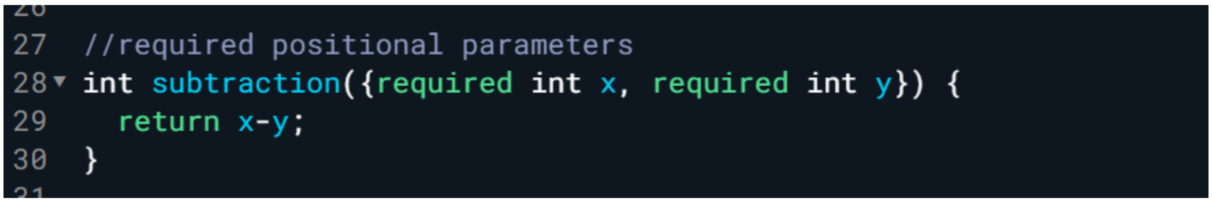


```
  //calling a function with positional parameters
  //which has its second argument as optional
  printNames("Elif","Selin");
  printNames("Ayşe");

```

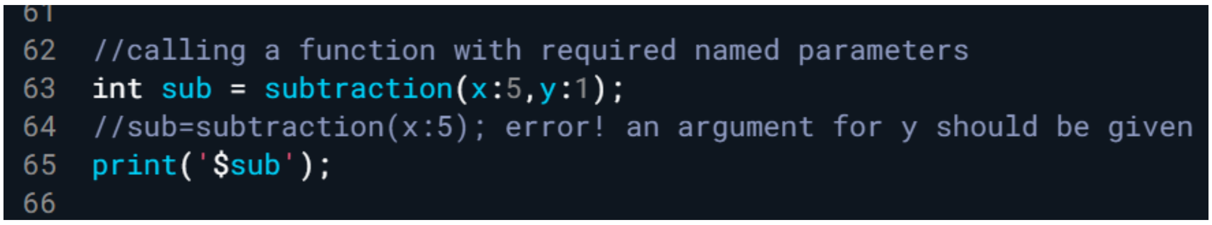


```
First name is: Elif
Second name is: Selin
First name is: Ayşe
```

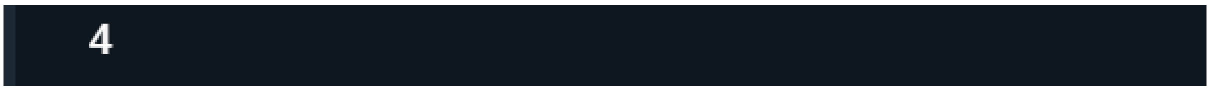
Named parameters are specified by assigning value to their names [4]. In Dart named parameters should be given in curly braces{} in the function signature. When called in the main, the parameter name should be followed by the : operator, which then is followed by the value of the named parameter [5].The example below shows a function called subtraction() with required named parameters. To make the parameters required, the *required* statement is used [5]. When the function is called from main, the value for both of the parameters, x and y, should be given. When only called with one argument, the example code below gives an error.



```
//required positional parameters
int subtraction({required int x, required int y}) {
  return x-y;
}
```

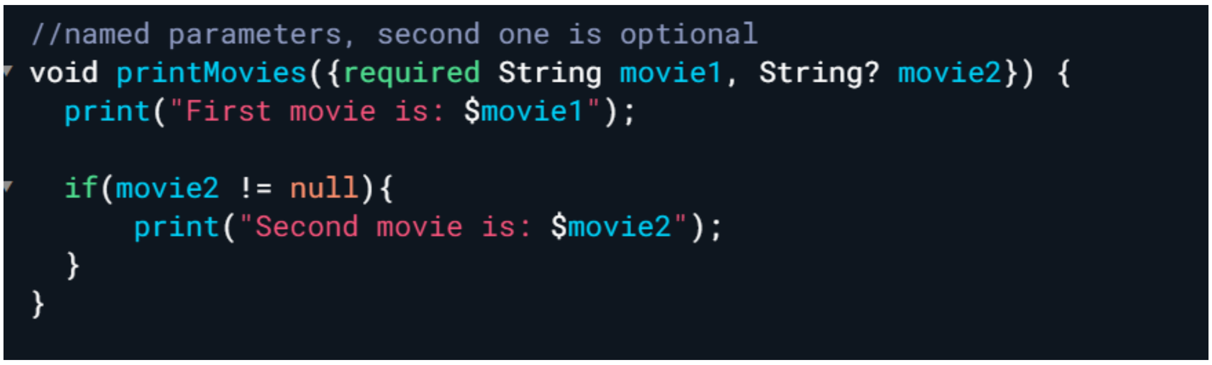


```
//calling a function with required named parameters
int sub = subtraction(x:5,y:1);
//sub=subtraction(x:5); error! an argument for y should be given
print('$sub');
```

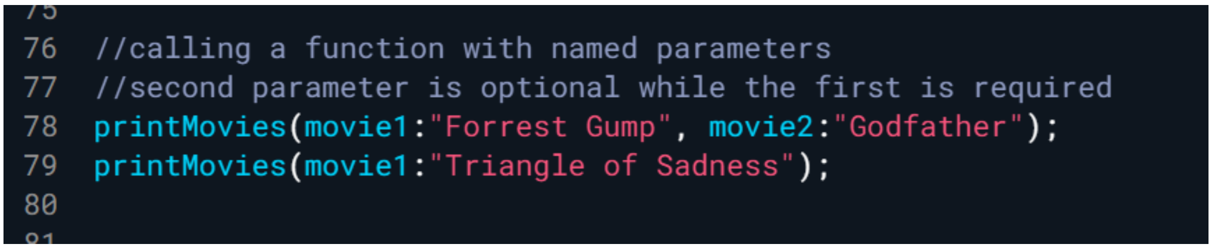


```
4
```

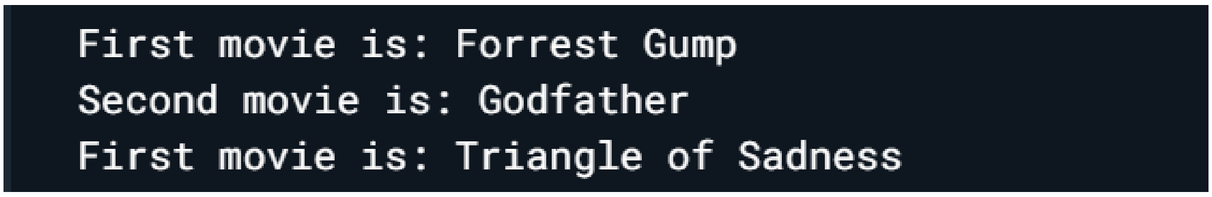
Named parameters can also be optional as well. When the *required* statement is not used in the function signature, they become optional [5]. In the example below, the parameter movie2 is optional. It can be seen that the function printMovies() can be called with a single argument in the main since the second one is optional.



```
//named parameters, second one is optional
void printMovies({required String movie1, String? movie2}) {
  print("First movie is: $movie1");

  if(movie2 != null){
      print("Second movie is: $movie2");
  }
}
```



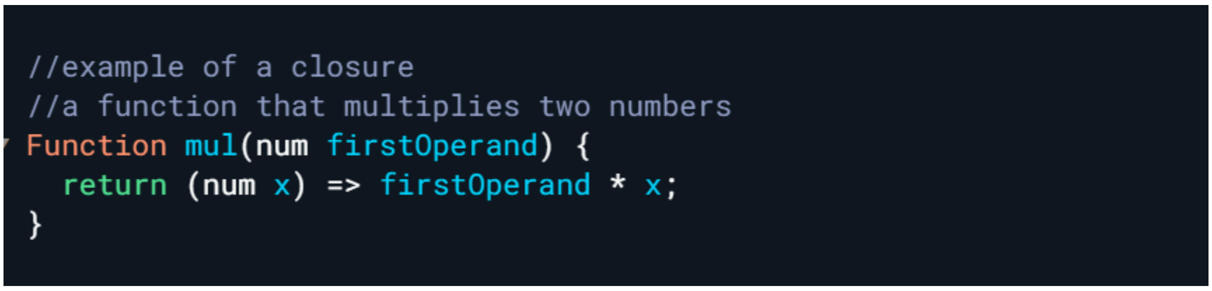
```
//calling a function with named parameters
//second parameter is optional while the first is required
printMovies(movie1:"Forrest Gump", movie2:"Godfather");
printMovies(movie1:"Triangle of Sadness");
```



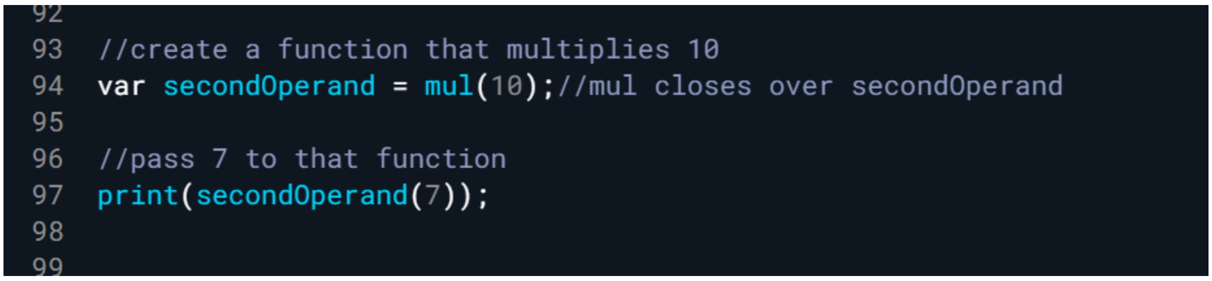
```
First movie is: Forrest Gump
Second movie is: Godfather
First movie is: Triangle of Sadness
```

5.Closures

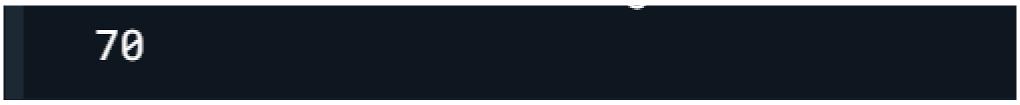
A closure is a function object that can access variables that are inside its lexical scope, even when the function is used outside of its original scope [6]. Dart language supports lexical closures. In the example below, the function mul() returns a function that multiplies firstOperand with the function's argument. In the main, variable secondOperand is a function that multiplies its argument with 10. It can be said that mul() closes over secondOperand. When the secondOperand is called with the argument 7, it multiplies 10 and 7 and returns 70. The function mul() captures the variable firstOperand meaning that wherever the returned function goes, it remembers firstOperand.



```
//example of a closure
//a function that multiplies two numbers
Function mul(num firstOperand) {
  return (num x) => firstOperand * x;
}
```



```
//create a function that multiplies 10
var secondOperand = mul(10);//mul closes over secondOperand

//pass 7 to that function
print(secondOperand(7));
```



```
70
```

6.Evaluation of the Languages

I think the nested subprograms, scope of local variables, parameter passing methods, parameters and closures were both easy to read and write in the Dart language. Dart allows nested subprograms which is an aid to readability. The fact that there is no pass by reference makes Dart more readable and writable. Named parameters in Dart helps with both readability and writability, even though the syntax of the required and optional parameters might be a little tricky. I think closures make the language less writable and, since the syntax of it is not so obvious. In conclusion, I believe, even though implementing some parts were tricky, the Dart language is readable and writable in terms of subprograms.

7.Learning Strategy

To learn and study nested subprograms, scope of local variables, parameter passing methods, parameters and closures in Dart, I first checked the official documentation. Then I tried writing the code, and when I had an error, I checked the internet for the actual syntax that should have been used. When I faced a specific error, I solved it by looking at stackoverflow and other forums. To study concepts such as closures, I checked our course book *Concepts of Programming Languages*. Even though I was not familiar with Dart, I did not have any difficulty examining the operations and subprograms in Dart, since I found its syntax to be easy and similar to Java.
The compiler that I have used is:

https://dartpad.dev/?

8. References

[1] "*How to create a nested function in Dart.*" geeksforgeeks.org
https://www.educative.io/answers/how-to-create-a-nested-function-in-dart (accessed December 12, 2022).
[2] "*Lexical Scope .*" dart.dev
https://dart.dev/guides/language/language-tour#lexical-scope (accessed December 12, 2022).
[3] "*Is Dart Pass by Reference?*." stackoverflow.com
https://stackoverflow.com/questions/58966090/is-dart-pass-by-reference#:~:text=Dart%20is%20only%20passed%20by,does%20not%20support%20reference%20passing. (accessed December 16, 2022).
[4] "*Positional vs. Named Parameters.*"java2s.com
http://www.java2s.com/Tutorial/CSharp/0200__Attribute/PositionalvsNamedParameters.htm (accessed December 14, 2022).
[5] "*Dart Parameters.*"
https://sarunw.com/posts/dart-parameters/ (accessed December 14, 2022).
[6] "*Lexical Closures .*" dart.dev
https://dart.dev/guides/language/language-tour#lexical-scope (accessed December 12, 2022).