CS315

Homework Assignment 3

Fall 2022

Zeynep Doğa Dellal 22002572 Section 3

Instructor: Karani Kardaş

Teaching Assistants:

Dilruba Sultan Haliloğlu, Hamza İslam, Onur Yıldırım

1. Nested subprogram definitions

The nested subprograms in Dart are defined as shown. To call inner function, outer function must be called first. Then outer function calls the inner function which is inside outer function.

```
int func(int x) {
  int y = 15;
  int innerfunc() {
    x = x * y;
    return x;
  }
  return innerfunc();
}
void main() {
  print(func(2));
}
```

Output: 30

2. Scope of local variables

The function fun shows an example for local variables in Dart. A local variable declared in a function cannot be reached from outside of the function. The function fun2 exemplify reaching a global variable inside a function. The function named func demonstrates usage of local and global variables. In nested loops, inner loop can reach the local variable of the outer loop but outer loop cannot reach the local variable of the outer loop.

```
void fun() {
  int xInFun = 5;
  print("Printing local variable in fun: $xInFun");
}

int x = 10;
void fun2() {
  print("Printing global variable x: $x");
}

void func() {
  int y = 15;
  print("Printing local variable in outer function: $y");
  void innerfunc() {
    int a = 20;
    print("Printing local variable in inner function: $a");
    print("Printing local variable of outer function in inner function: $y");
    y = 3;
    print("Printing local variable of outer function in inner function after changing: $y");
  }
  innerfunc();
  print("Printing local variable of outer function in outer function after changing: $y");
  //print("Printing local variable in innerfunc from outer func: $a");
}

void main() {
  fun();
  fun2();
  func();
}
```

Output:

Printing local variable in fun: 5

Printing global variable x: 10

Printing local variable in outer function: 15

Printing local variable in inner function: 20

Printing local variable of outer function in inner function: 15

Printing local variable of outer function in inner function after changing: 3

Printing local variable of outer function in outer function after changing: 3

3. Parameter passing methods

Dart supports positional parameters. Type checking exists in Dart. If the type of header parameters and actual parameters don't match, compiler shows error. Dart also supports passing functions as parameters. When values are passed to functions, values are not copied. Changes that are done to passed by value inside the function is only viable inside that function.

```
int power(int x, [int y = 4]) {
  int a = 1;
  for (int i = 0; i < y; i++) {
    a = a * x;
  }
  return a;
}

int increment(int x) => ++x;

int apply(int x, Function f) {
  return f(x);
}

void sub(int x){
  x--;
  print("Inside of function value is: $x");
}

void main() {
  int x = 10;
  print(power(2, 5));
  print(apply(12, increment)); //pass function
  print("Outside of function before apply the x value is: $x");
  sub(x);
  print("Outside of function after apply the x value is: $x");
}
```

Output:

32

13

Outside of function before apply the x value is: 10

Inside of function value is: 9

Outside of function after apply the x value is: 10

4. Keyword and default parameters

Dart does not support keyword parameters and the default parameters must appear last because parameters are positionally associated. Program does not compile if all parameters are default parameters and there are no passed value while calling the function.

```
int power(int x, [int y = 4]) {
  int a = 1;
  for (int i = 0; i < y; i++) {
    a = a * x;
  }
  return a;
}


void main() {
  print(power(2));
  //print(power());
}
```

Output:

16

5. Closures

In Dart, closures are similar to functions as they can take parameters and return a value. Functions have the ability to close over variables created in surrounding scopes.

```
Function makePow(int multiplyBy) {
  return (int changable) => multiplyBy * changable;
}

void main(){
  var mul2 = makePow(2);


  var mul15 = makePow(15);

  int a = mul2(15);
  int b = mul15(15);
  print("15 multiplied by 2: $a");
  print("15 multiplied by 15: $b");
}
```

Output:

15 multiplied by 2: 30

15 multiplied by 15: 225

6. Evaluation of Dart Language

Dart language is easy to use in terms of subprograms and does support necessary features to support like nested functions, type checking, closures, default parameters. In terms of readability Dart is sufficient because it has easily understandable function names and it defines the scope of subprograms using "{}". In terms of writability only confusing aspect of Dart is the requirement of "$" character before using a variable while printing. It makes it hard to write Dart. Even though extra symbols such as "{}, [],; ()" are hard to write, they make Dart language easier to read and understand therefore in my opinion they are necessary.

7. Learning Strategy

I already knew how to declare variables, output the results on the console and the basics of Dart from previous homeworks. I looked into whether Dart supports nested functions, how scope of variables work in Dart, whether it supports keyword or positional parameters, whether it supports default parameters and how to use closures. I experimented whether Dart supports the same features I know and use comfortably like Java and C++. I used the following resources:

Dart official documentation site: https://dart.dev/

Dart compiler: https://dartpad.dev/

Dart examples that I used to learn closures: https://sites.google.com/site/dartlangexamples/learn/

Some additional sites that helped: https://www.tutorialspoint.com/index.htm

Some additional sites that helped: https://www.geeksforgeeks.org/

8. Communication

This homework was relatively easier than others. In other homeworks I had discussions with my friends on how to do the assignments but in this homework only thing we talked about was how long it took to do this homework.