605.629: Programming Languages
Assignment 8
Sabbir Ahmed

October 31, 2021

1. [20 pts, Subprograms]

Imagine you need to write a function which prints an integer value on the screen. This value represents a counter and is set to '1' initially and then incremented on subsequent calls to the same function. Can it be done in Java without using function parameters or non-local variables? How about a C macro?

Answer

Achieving this counter would not be possible as a C macro since the preprocessor would expand the counter to 1 and then pass it to the compiler. The macro then becomes an rvalue, and therefore cannot be further modified. It is not possible to achieve this in Java using non-local variables, The counter would have to be member variable outside the function so that it would not reset its value when the function reaches the end of its scope.

---

2. [40 pts] Consider the following code slice using Java 8 Function interface.

```
import java.util.function.Function;

public class Test1 {

    public Function<Integer, String> getTextOfWeekday = num -> {
        String[] weeks = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
        return (num > 0 && num <= weeks.length) ? weeks[num - 1] : null;
    };

    public void test() {
        System.out.println(getTextOfWeekday.apply(3));
    }

    public static void main(String args[]) {
        Test1 t = new Test1();
        t.test();
    }
}
```

Output: `Wed`

a. What is (are) the name of concept(s) this code slice use (that we have seen in this course)?

Answer

One of the concepts used in this code slice is functional programming, demonstrated through the lambda function `getTextOfWeekday`.

---

b. Write another Function TestPrint and print the same result using "`TestPrint o getTextOfWeekday`" in order to demonstrate the concept that you specified in (a.) The output should be same and the form of the call should be (in generic terms) `function1.apply(function2.apply(...))`.

Answer

```
import java.util.function.Function;

public class Test2 {

    public Function<Integer, String> getTextOfWeekday = num -> {
        String[] weeks = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
        return (num > 0 && num <= weeks.length) ? weeks[num - 1] : null;
    };

    public Function<String, String> testPrint = textOfWeekday -> {
        return textOfWeekday;
    };

    public void test() {
        System.out.println(getTextOfWeekday.apply(3));
        System.out.println(testPrint.apply(getTextOfWeekday.apply(3)));
    }

    public static void main(String args[]) {
        Test2 t = new Test2();
        t.test();
    }
}
```

`testFunction` introduces another concept in functional programming, which is function composition.

---

3. [40 pts] Write a tail recursive factorial (in a language of your choice) and show the activation record for 'factorial of 5'.

Answer

[1]:
```
def fact(n, k = 1):

    if (n == 1):
        return k

    return fact(n - 1, n * k)

print(fact(5))
```

```
120
```

`fact(5, 1)`	
control	
return val	
n	5
k	1

1.

`fact(4, 5)`	
control	
return val	
n	4
k	5

2.

`fact(3, 20)`	
control	
return val	
n	3
k	20

3.

`fact(2, 60)`	
control	
return val	
n	2
k	60

4.

`fact(1, 120)`	
control	
return val	
n	1
k	120

5.