CS 61B
Spring 2021

# Functional Programming

Topical Session 0: January 31, 2021

## 1 Be My Main

The following code is given in Python.

```
def assign_grade(grade):
    if grade < 60:
        print('F')
    elif grade < 70:
        print('D')
    elif grade < 80:
        print('C')
    elif grade < 90:
        print('B')
    else:
        print('A')
```

(a) Convert the above code to Java. Keep in mind differences in syntax, structure, and naming conventions.

Solution:

```
public class Grading {
    public static void assignGrade(int grade) {
        if (grade < 60) {
            System.out.println('F');
        } else if (grade < 70) {
            System.out.println('D');
        } else if (grade < 80) {
            System.out.println('C');
        } else if (grade < 90) {
            System.out.println('B');
        } else {
            System.out.println('A');
        }
    }
}
```

(b) Define each word in the `main()` method

```
public static void main(String[] args)
```

**public**: Public is an access modifier, which is used to specify what can access this method. Public means that this method will be accessible by any class. Other access modifiers include private (only the current class can access it) and protected (only classes in the same package can access it).

**static**: A keyword in java which identifies that it is class-based. It is made static in Java so that it can be accessed without creating the instance of a class. This is especially useful for a main function since it is the first thing that is run, so it must be runnable without already having access to a class instance.

**void**: The return type of the method. Void means that the method will not return any value.

**main**: The name of this function. In this case, the name is special because our computer looks for a function named main when executing a class.

**String args[]**: The parameters being passed in to the function, in this case an array of Strings stored in a variable called args. For the main function, this input comes from command line arguments.

## 2 Two Sum

Given a list of numbers, find two numbers such that their adding them would result in the value `target`. Assume that only one such pair exists, and there is always a pair that fulfills the condition. Return the numbers as an array with two terms.

Solution:

```
int[] twoSum(int[] nums, int target) {
    for(int i = 0; i < nums.length; i += 1) {
        int complement = target - nums[i];
        for (int j = 0; j < nums.length; j += 1) {
            if (complement == nums[j] && i != j) {
                int[] result = {nums[i], nums[j]};
                return result;
            }
        }
    }
}
```

## 3 Skipping Fibonacci

In the standard Fibonacci sequence, we add the previous two numbers of a sequence to create the next number of the sequence. This can also be represented as adding the (n-1)th term and the (n-2)th term.

$$0, 1, 1, 2, 3, 5, 8, 13, 21...$$

In this skipping fibonacci version, we add the (n-1)th term and the (n-3)th term to get the next number of the sequence.

$$0, 1, 1, 1, 2, 3, 4, 6, 9, 13, 19...$$

To get the 4th term, we add the 3rd and 1st term. $f(4) = f(3) + f(1) = 1 + 1 = 2$. Similarly, to get the 5th term, we add the 4th and 2nd term. $f(5) = f(4) + f(2) = 2 + 1 = 3$. Write an algorithm that would return the nth term of the skipping fibonacci sequence.

Solution:

```
int skiponacci(int n) {
    if (n == 0) {
        return 0;
    }
    else if (n == 1 || n == 2) {
        return 1;
    } else {
        return skiponacci(n - 1) + skiponacci(n - 3);
    }
}
```

## 4 Let's Get That Bread

Over the summer, you decide to take multiple jobs. Each of these jobs takes a certain amount of time, and pays a certain rate. In your notebook, you write down how many hours you worked and how much money you earned per hour for that week. You want to make a class that allows you to add items to your journal and get your total earnings whenever you want. Given a list of hours worked, and another list of per-hour payment rates, write `payCalc` to return the total money earned.

You can assume that the arrays `wages` and `hours` are of equal length.

```
public class Journal {
    private int[] wages;
    private int[] hours;

    public Journal(int[] wages, int[] hours) {
        this.wages = wages;
        this.hours = hours;
    }

    public void addEntry(int wages, int hours) {
        // Assume this is implemented
    }

    public int payCalc() {
        int total = 0
        for (int i = 0; i < wages.length; i += 1) {
            total += wages[i] * hours[i];
        }
        return total;
    }
}
```

## 5 Add Digits Add Digits Add Digits A—

Given a number, add its digits together to get another number. Keep doing this until the result is less than 10, then return the value. For example,

$$12349 \xrightarrow{1+2+3+4+9} 19 \xrightarrow{1+9} 10 \xrightarrow{1+0} 1$$

Hint: Think about how you can use both recursion and iteration to tackle this question!

Solution:

```java
public int addDigits(int n) {
    if (n < 10) {
        return n;
    }
    else {
        int sum = 0;
        while (n != 0) {
            sum += n % 10;
            n = n / 10;
        }
        return addDigits(sum);
    }
}
```

## 6 The Reptile Room

Write out the what the program will output.

```java
public class Reptile {
    public String type;
    public String name;
    public static String location;
    public int age;

    public Reptile(String type, String name, String location, int age) {
        this.type = type;
        this.name = name;
        this.location = location;
        this.age = age;
    }

    public static void relocate(String l) {
        location = l;
    }

    public static void birthday(Reptile a) {
        a.age += 1;
    }

    public static void swap(Reptile a, Reptile b) {
        String temp = a.type;
        a.type = b.type;
        b.type = temp;
    }

    public static void flop(Reptile a, Reptile b) {
        Reptile temp = a;
        a = b;
        b = temp;
    }

    public static void main(String[] args) {
        Reptile a = new Reptile("Iguana", "Isabella", "North Carolina", 3);
        Reptile b = new Reptile("Snake", "Katya", "Colorado", 5);
        System.out.println(a.location);
        Reptile c = new Reptile("Crocodile", "Suha", "California", 1);
        System.out.println(a.location);
        System.out.println(Reptile.location);
        Reptile d = new Reptile("Gator", "Ram", "Georgia", 6);
        System.out.println(b.location);
        relocate("Alaska");
        System.out.println(c.location);
        System.out.println(d.location);
        System.out.println(Reptile.location);
        birthday(a);
        System.out.println(a.name+" the "+a.type+" turned "+String.valueOf(a.age)+" in "+a.location+"
    !");
        flop(c, b);
        System.out.println(b.type);
        swap(d, c);
        System.out.println(d.type);
    }

}
```

```
Colorado
California
California
Georgia
Alaska
Alaska
Alaska
Isabella the Iguana turned 4 in Alaska!
Snake
Crocodile
```