CSE 2102: Introduction to Software Engineering
Lab #3: February 8, 2022
Conditionals, Loops, Arrays

Note: Lab assignments are intended for practice. These will not be graded, and need not be submitted.

A. Conditionals and Loops

Conditionals allow you to change the control flow, or the sequence in which the statements in a program are executed. Looping constructs, allow the execution of a certain group of statements multiple times depending on certain conditions. Java supports the common conditional and looping constructs, namely, `if-then-else`, `while` and `for`. In using these constructs, the scope of a variable, the body of the code where a variable is accessible, must be considered. The control flow structures in Java are identical to those used in other programming languages.

Consider the following program that takes a 9-digit integer as a command-line argument, computes the checksum, and prints the 10-digit ISBN number. The ISBN is a 10-digit code that uniquely identifies a book. The rightmost digit is a checksum digit which can be uniquely determined from the other 9 digits from the condition that **d1 + 2*d2 + 3*d3 + ... + 10*d10** must be a multiple of 11 (here, di denotes the i-th digit from the right. For instance, d1 is the rightmost digit and d10 is the leftmost digit). The checksum digit d1 can be any value from 0 to 10 (inclusive). The ISBN convention is to use the value X to denote 10. Example: the checksum digit corresponding to 020131452 is 5 since it is the only value of d1 between 0 and 10 where the resulting value **of d1 + 2*2 + 3*5 + 4*4 + 5*1 + 6*3 + 7*1 + 8*0 + 9*2 + 10*0** is a multiple of 11.

ISBN.java ArrayTest.java

```java
public class ISBN {
    public static void main(String[] args) {

        // read in one command-line argument
        int n = Integer.parseInt(args[0]);

        // compute the weighted sum of the digits, from right to left
        int sum = 0;
        for (int i = 2; i <= 10; i++) {
            int digit = n % 10;         // rightmost digit
            sum = sum + i * digit;
            n = n / 10;
        }

        // print out check digit, use X for 10
        System.out.println("The full ISBN is " + args[0]);
        System.out.print("The checksum is ");
        if      (sum % 11 == 1) System.out.println("X");
        else if (sum % 11 == 0) System.out.println("0");
        else                    System.out.println(11 - (sum % 11));
    }
}
```

B. Arrays

Programming an array in Java requires three distinct steps: a) Declare the array name, b) Create the array, and c) Initialize the array values. The following code snippet shows the three steps:

```java
double[] a;                     // declare the array
a = new double[n];              // create the array
for (int i = 0; i < n; i++) {   // elements are indexed from 0 to n-1
    a[i] = 0.0;                 // initialize all elements 0.0
}
```

Once we create an array, its length is fixed. You can refer to the length of an `a[]` in your program with the code `a.length`. We can take advantage of Java's default array initialization convention. For example, the following statement is equivalent to the above four lines:

```java
double[] a = new double[n];
```

The statement above will define the array `a` with type `double[]` and set it equal to a double array with length `n`. The default initial value is 0 for all numeric primitive types and false for type Boolean. We can also set the array values at compile time, as shown in the following code snippet that processes playing cards, and then prints a random card name.

```
String[] SUITS = {"Clubs", "Diamonds", "Hearts", "Spades"};
String[] RANKS = {"2", "3", "4", "5", "6", "7", "8", "9", "10",
        "Jack", "Queen", "King", "Ace"};

int i = (int) (Math.random() * RANKS.length);
int j = (int) (Math.random() * SUITS.length);
System.out.println(RANKS[i] + " of " + SUITS[j]);
```

We may also wish to compute and assign values to the elements of the array at run time. For example, the following code spinet initializes an array of length 52 that represents a deck of playing cards, using the arrays `RANKS[]` and `SUITS[]` defined above. Notice carefully how the indices are used in the for loops and assignment operations in the code snippet.

```
String[] deck = new String[RANKS.length * SUITS.length];
for (int i = 0; i < RANKS.length; i++) {
    for (int j = 0; j < SUITS.length; j++) {
        deck[SUITS.length*i + j] = RANKS[i] + " of " + SUITS[j];
    }
}
```