# Assignment 3

Enrique Saracho Felix
100406980
CPSC 1150
01/07/2023

## Exercise 1

### Program Count

**File name:** Count.java

**Purpose:** To prompt the user to enter a string, and count and display the number of uppercase letters and digits in it.

**Packages:** java.util.Scanner

**Limitations:** The input must be of type String.

**Input:** A string, stored in the variable *frase*.

**Output:** A message containing the number of uppercase letters and digits.

**Pseudocode:**

```
Algorithm Count
START

(main)
Set string frase
Read frase
Print “The number of uppercase letters is “ + countUpperCase( frase )
Print “The number of digits is “ + countDigits( frase )

(countUpperCase, parameters: string frase)
Set count = 0
For i in frase
{
        If frase[ i ] >= 65 && frase[ i ] <= 90 then
        {
                count++
        }
}
Return count

(countDigits, parameters: string frase)
Set count = 0
For i in frase
{
        If frase[ i ] >= 48 && frase[ i ] <= 57 then
```

```
        {
                count++
        }
}
```
Return *count*

END *Count*

**Test run(s):**

```
$ java Count.java
Enter a string: Welcome to Java 8 (1.8)
The number of uppercase letters is 2
The number of digits is 3
```

```
$ java Count.java
Enter a string: Hello World!
The number of uppercase letters is 2
The number of digits is 0
```

```
$ java Count.java
Enter a string: Ram 1500
The number of uppercase letters is 1
The number of digits is 4
```

# Exercise 2

## Program SSN

**File name:** SSN/java

**Purpose:** To validate the format of a Social Security Number entered by the user.

**Packages:** java.util.Scanner

**Input:** A social security number in the format DDD-DDD-DDD. The value is stored in a variable of type String.

**Output:** A message informing the user if the format is correct or not.

**Pseudocode:**

Algorithm SNN
START

(*main*)
Declare String *ssn*
Read *ssn*
If ( *validation1*(*ssn*) && *validation2*(*ssn*) && *validation3*(*ssn*) && *validation4*(*ssn*) && *validation5*(*ssn*) ) is true then
{
        Print *ssn* + " is a valid social security number"
}
Else

```
{
        Print ssn + " is an invalid social security number"
}

(validation1, parameters: String ssn)
If ( ssn.length != 11 ) is true then
{
        Return false
}

(validation2, parameters: String ssn)
For i in ssn
{
        If ( i = 3 || i = 7 ) is true then
        {
                Jump iteration
        }
        If ( ssn[ i ] < 48 || ssn[ i ] > 57 ) is true then
        {
                Return false
        }
}

(validation3, parameters: String ssn)
If ( ssn[ 3 ] != "-" || ssn[ 7 ] != "-" ) is true then
{
        Return false
}
Return true

(validation4, parameters: String ssn)
If ( ssn[ 0 ] = "0" ) is true then
{
        Return false
}
Return true

(validation5, parameters: String ssn)
If ( ssn[ 4 ] < "1" ) is true then
{
        Return false
}
If (ssn[ 4 ] = "1" && ssn[ 5 ] = "0" && ssn[ 6 ] = "0"  )
{
        Return false
}
Return true
```

END SNN

**Test run(s):**

```
$ java SSN.java
Enter a SSN: 123-268-097
123-268-097 is a valid social security number
```

```
$ java SSN.java
Enter a SSN: 023-289-097
023-289-097 is an invalid social security number
```

```
$ java SSN.java
Enter a SSN: 198-068-097
198-068-097 is an invalid social security number
```

```
$ java SSN.java
Enter a SSN: 198-1680-97
198-1680-97 is an invalid social security number
```

# Exercise 3

## Program PalindromePrime

**File name:** PalindromePrime.java

**Purpose:** To print the first 100 palindromic prime numbers in tabular format.

**Input:** Not needed.

**Output:** A 10x10 table displaying the first 100 numbers that are both prime and palindrome.

**Pseudocode:**

Algorithm (program name)
START

(*main*)
Set integer *count* = 100
Set integer *lineCount* = 0
Set integer *num* = 2
While ( *count* > 0 )
{
    While ( *lineCount* < 10 )
    {
        If ( *isPrime*(*num*) && *isPalindrome*(*num*) ) is true then
        {
            Print *num*
            *count*--
            *lineCount*++
        }
        *num*++

```
        }
        Print new line
        lineCount = 0
}

(isPrime, parameters: integer num)
Set integer i = 2
While ( i  < num )
{
        If ( num % i == 0 ) is true then
        {
                Return false
        }
        i++
}
Return true

(isPalindrome, parameters: integer num)
Set integer normalNum = num
Set integer reverseNum = 0
While ( normalNum > 0 )
{
        reverseNum = (reverseNum * 10 + normalNum) % 10
        normalNum /= 10
}
If ( reverseNum = num ) is true then
{
        Return true
}
Return false

END (program name)
```

**Test run(s):**

```
$ java PalindromePrime.java
2     3     5     7     11    101   131   151   181   191
313   353   373   383   727   757   787   797   919   929
10301 10501 10601 11311 11411 12421 12721 12821 13331 13831
13931 14341 14741 15451 15551 16061 16361 16561 16661 17471
17971 18181 18481 19391 19891 19991 30103 30203 30403 30703
30803 31013 31513 32323 32423 33533 34543 34843 35053 35153
35353 35753 36263 36563 37273 37573 38083 38183 38783 39293
70207 70507 70607 71317 71917 72227 72727 73037 73237 73637
74047 74747 75557 76367 76667 77377 77477 77977 78487 78787
78887 79397 79697 79997 90709 91019 93139 93239 93739 94049
```

# Exercise 4

## Program RSPGame

**File name:** RSPGame.java

**Purpose:** To allow the user to play rock, scissors, paper game against the computer to the best of 5.

**Packages:** java.util.Scanner

**Limitations:** The program will display an error message if the input is less than 0 or more than 2. And will prompt again until the input is correct.

**Input:** An integer number between 0 and 2 (inclusive). To represent the shape selected.

**Output:** A string message informing the user of the result of the game.

**Pseudocode:**

```
Algorithm RSPGame
START

(main)
playRSP()

(playRSP)
Set integer userWins = 0
Set integer compWins = 0
Set integer rount = 1
Set string result
While ( round <= 5 && userWins < 3 && compWins < 3 ) is true
{
        result = playRound()
        if ( result = 1 )
        {
                userWins++
        }
        Else if ( result = 2 )
        {
                compWins++
        }
        round++
}
displayResult( userWins, compWins )

(playRound)
Set integer userNum
Set integer compNum
Set integer result
userNum = getUserInput()
```

```
compNum = getCompInput
result = compareNums( userNum, compNum )
return result

(compareNums, parameters: userNum, compNum)
If ( userNum = compNum ) is true then
{
        Print draw message
        Return 0
}
Else if ( userNum = 0 ) is true then
{
        If ( compNum = 1 ) is true then
        {
                Print user wins
                Return 1
        }
        Else then
        {
                Print computer wins
                Return 2
        }
}
Else if ( userNum = 1 ) is true then
{
        If ( compNum = 2 ) is true then
        {
                Print user wins
                Return 1
        }
        Else then
        {
                Print computer wins
                Return 2
        }
}
Else then
{
        If ( compNum = 0 ) is true then
        {
                Print user wins
                Return 1
        }
        Else then
        {
                Print computer wins
                Return 2
```

```
        }
}
```

(*getUserInput*)
Set integer userInput
Set boolean flag = false
Do
{
        If ( flag ) is true then
        {
                Print error message
        }
        Read userInput
        flag = true
} While ( userInput < 0 || userInput > 2 )
Return userInput

(*getCompInput*)
Return random[0, 1, 2]

(*displayResult*, parameters: *userWins compWins*)
If ( userWins > compWins ) is true then
{
        Print user wins!
}
Else then
{
        Print computer wins!
}

END RSPGame

**Test run(s):**

```
$ java RSPGame.java
Select rock(0), scissor(1), or paper(2): 1
The computer is scissor. You are scissor too. It is a draw.
Select rock(0), scissor(1), or paper(2): 2
The computer is paper. You are paper too. It is a draw.
Select rock(0), scissor(1), or paper(2): 3
Error: invalid input. Try again
Select rock(0), scissor(1), or paper(2): 0
The computer is scissor. You are rock. You won!
Select rock(0), scissor(1), or paper(2): 2
The computer is rock. You are paper. You won!
Select rock(0), scissor(1), or paper(2): 2
The computer is paper. You are paper too. It is a draw.
Game over! You are the winner!
```

```
$ java RSPGame.java
Select rock(0), scissor(1), or paper(2): 1
The computer is rock. You are scissor. You lose.
Select rock(0), scissor(1), or paper(2): 5
Error: invalid input. Try again
Select rock(0), scissor(1), or paper(2): 4
Error: invalid input. Try again
Select rock(0), scissor(1), or paper(2): 0
The computer is rock. You are rock too. It is a draw.
Select rock(0), scissor(1), or paper(2): 1
The computer is scissor. You are scissor too. It is a draw.
Select rock(0), scissor(1), or paper(2): 0
The computer is scissor. You are rock. You won!
Select rock(0), scissor(1), or paper(2): 0
The computer is scissor. You are rock. You won!
Game over! You are the winner!
```

```
$ java RSPGame.java
Select rock(0), scissor(1), or paper(2): 1
The computer is paper. You are scissor. You won!
Select rock(0), scissor(1), or paper(2): 2
The computer is rock. You are paper. You won!
Select rock(0), scissor(1), or paper(2): 2
The computer is scissor. You are paper. You lose
Select rock(0), scissor(1), or paper(2): 1
The computer is rock. You are scissor. You lose.
Select rock(0), scissor(1), or paper(2): 0
The computer is paper. You are rock. You lose.
Game over! The Computer is the winner! Try again!
```