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):**

A black screen with white text

Description automatically generated with low confidence

A black screen with white text

Description automatically generated with low confidence

A black background with white text

Description automatically generated with low confidence

# 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):**

A picture containing text, font, screenshot, typography

Description automatically generated

A black background with white text

Description automatically generated with low confidence

A black background with white text

Description automatically generated with low confidence

A black background with white text

Description automatically generated with low confidence

# 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):**

A screenshot of a computer screen

Description automatically generated with medium confidence

# Q1 – Question 1

## Program (program name)

**File name:** (file’s relative path)

**Purpose:** To have a sample from which to create external documentation.

**Packages:** (list of imported packages)

**Limitations:** (input it can’t handle, list of possible error messages, round-off error)

**Bugs:** (list of unfixed bugs)

**Input:** …

**Output:** …

**Pseudocode:**

Algorithm (program name)

START

1. Step 1
2. .
3. .
4. .

END (program name)

**Test run(s):**