*Report: Week 2: Assignment 1 – BinarySum.java*

Take two binary numbers as inputs for the program

Check whether the inputted numbers of the correct binary format.

Print an error message and prompt the user to input the correct number format.

Pass the numbers to the function which converts the binary to decimal number

Calculate the sum of the two decimal numbers.

Convert the summation result to binary and hexadecimal number format.

Print the sum result in decimal, binary and hexadecimal formats.

NO

YES

(1)

(2)

(3)

(4)

(5)

(6)

The problem we have to implement a program that acts as a calculator for adding of two numbers, these numbers should be in the binary number format. The basic idea we have adopted – which we feel is less complicated – is, by converting the inputted binary numbers into a decimal number and then adding those numbers to get the sum. The flowchart above gives a brief outline of how the program has been instrumented. As shown in the step (1), exactly two binary numbers are taken as an input, if there are more than or less than two command line arguments given the program prints out a message saying to input the correct number of arguments. These binary numbers are then check – as shown in step (2) - for their correct number format and are passed to the function which converts them to decimal number format [step (3)]. The sum of these numbers is calculated [step (4)] and then the final sum is converted to a binary and hexadecimal number format – as shown in step (5). Ultimately the desired output of the sum in all the number format is printed.

The initial idea was to implement the code by directly calculating the sum of the two binary numbers and then converting it into the desired number format. But this method would have required additional computations and methods for it.

*Report: Week 2: Assignment 2 – StringThing.java*

***equalityTest1***function --

abcV2

abcV7

String Intern Pool

String Intern Pool: String Intern Pool is a separate location in memory where java stores all unique String literals created in the code. When double quotes are used to create String, JVM first looks for the same value in the String pool, if found it just returns the reference else it creates a new String in the pool and returns the reference. However using new operator, we force String class to create a new String object in heap memory.

abcV2:

Since abcV2 is made by concatenating “a”+”b”+”c” and hence its String literal. Therefore, == points to the same memory address in String pool for abcV2 and abcV1 which shows the output true.

abcV3:

abcV3 is made by using substring function and hence it’s abcV3 is computed at Run time. This results in JVM storing abcV3 in heap memory. Therefore, == return false as abcV3 and abcV1 are in 2 different memory locations.

abcV4

Same as abcV3, abcV4 is also computed at runtime and hence == return false.

abcV5

abcV5 is computed to ab99 which is not equal to abcV1.

abcV6

As abcV6 is made by using new keyword, its stored in heap memory by JVM. As said above, == compares the memory location, the output of this statement is false.

abcV7

When the intern() method is invoked on a String object, JVM looks the String contained by

This String object in the intern pool. If a matching String is found in the intern pool, then the String from the pool is returned else, this String object is added to the pool and a reference to it is returned. And hence, abcV7== abcV1 is true.

abcV8

As abcV8 is same as abcV6, it is also stored in heap memory and not the String intern pool. Hence the output is false.

***equalityTest2***function --

This function uses .equals() method. And hence contents of 2 string will be compared and if it’s same, true is returned or else false.

True : abcV2, abcV3, abcV4, abcV6, abcV7, abcV8

False: abcV5

***equalityTest3***function –

In identityHashCode method the content are not taken in consideration, and hence, identityHashCode(obj) will be different for 2 different Strings even if they hold same value but hashcode will return the same hashcode in this case.

True: abcV2,abcV7

False : abcV3, abcV4, abcV5, abcV6, abcV8

***equalityTest4***function –

True:abcV2,abcV7,abcV3, abcV4,abcV6, abcV8

False:abcV5

***equalityTest5***function –

hashCode() calculates the hashcode of the content in the string and hence if 2 differnt Strings has same content, then hashcode will be same

True:All except abcV5