

1. preAndPostDifference()
   1. declare two local variables x and y initialised to 5 and 10 respectively
   2. output the value of x preceded by “The value of x is “
   3. output the value of ++x preceded by “The value of ++x is “
      1. System.out.println(“The value of ++x is “ + ++x );
   4. output the value of x++ preceded by “The value of x++ is “
      1. System.out.println(“The value of x++ is “ + x++ );
   5. output the value of x preceded by “The value of x is “
   6. the values output for x should be 5, 6, 6 and 7
   7. do the exact same for y, except use the prefix/postfix decrement operator i.e. --
   8. the values output for y should be 10, 9, 9 and 8.
2. booleanLogicShortCircuitOps()
   1. ask the user to enter 2 numbers, store in int variables num1 and num2 respectively
   2. code an *if* statement as follows:
      1. if(num1 < 0 && num2++ < 0){ /\* output num2 \*/ }  
         else if(num1 > 0 && num2++ > 0) { /\* output num2 \*/ }  
         else if (num1 == 0 || num2++ == 0) { /\* output num2 \*/ }  
         else if (num1 < 0 || num2++ < 0) { /\* output num2 \*/ }
      2. output num2
      3. test the logic with various values and understand why num2 has the value it has
3. booleanBitwiseOps()
   1. the logic is identical to the booleanLogicShortCircuitOps() except that the operators are NOT the short-circuit ones i.e. use & instead of && and | instead of ||
   2. test with the same values as in booleanLogicShortCircuitOps(). Note the difference in the value for num2 and understand why this is the case.
4. compareStrings()
   1. using Scanner, get in two strings from the user, namely s1 and s2. Input “Mary” for both.
   2. code the following System.out.println(“Using == returns “ + s1 == s2);
      1. the output should be “Using == returns false”
         1. what is output? why? can you fix it?
   3. code the following System.out.println(“Using equals() returns “ + s1.equals(s2));
   4. now we will use string literals and observer the difference:
      1. declare a String variable name1 and initialise it to “Sean”
      2. declare another String variable name2 and initialise it also to “Sean”
      3. compare them using ==
         1. what is the output? why?
   5. now we will use a String literal and a *new* string
      1. declare a String variable name3 and initialise it to “John”
      2. declare a String variable name4 using *new* and initialise it to “John”
         1. String name4 = new String(“John”);
      3. compare them using ==
         1. what is the output? why?
5. admitToFilm()
   1. in main() ask the user for the certification for a film (an int) and the age of the person seeking admittance (also an int)
   2. pass these two values down to admitToFilm() and output the boolean returned
   3. in admitToFilm() do the following:
      1. using a ternary operator check if the customer is old enough to see the film. Return true or false depending on the result.
6. ternaryOperator()
   1. declare a local boolean variable named *isHappy* and initialise it to *true*.
   2. using a ternary operator, initialise a String variable namely *mood* based on *isHappy* :
      1. if *isHappy* is *true* set *mood* to “ I am happy!” otherwise set it to “I am not happy!”
   3. ask the user for two int variables namely *x* and *y*
   4. using a ternary operator, initialise an *int* variable *minVal* to the minimum of *x* and *y* (assume that they are not equal).
   5. output *minVal*.
7. switchVowelOrConsonant()
   1. ask the user to enter a character (use *sc.next().charAt(0);* to retrieve the char from the keyboard. Store the char in a variable named *letter.*
   2. switch on *letter*
      1. if it’s a vowel (uppercase or lowercase) then output the letter entered followed by the text “ is a vowel”.
      2. otherwise output the letter entered followed by the text “ is a consonant”.
   3. change the uppercase letter case labels to use their Unicode/ASCII values i.e. instead of *case ‘A’:* use *case 65:*
   4. move the *default* section to the top of the *switch* statement. Does it still work properly?
8. ifMonth()
   1. prompt the user to enter a month in the range 1..12
   2. initialise *month* with the users input
   3. using an *if-else if-else* structure, output the text for the month entered i.e. if the user entered 1 output “January”; if the value is out of range, output the erroneous value followed by the text “ is out of range”.
   4. instead of using integer literals e.g. 1, 2, 3 etc.. use constants e.g. JAN, FEB, MAR. To do this, declare constants: *final int JAN=1, FEB=2, MAR=3;*
   5. wrap the whole *if* statement in a *try-catch* block; thus, if the user enters “abc” for the month, you will be able to cater for that. In the *catch* block, output “Exception caught!” and print the stack trace.
9. ifGrade()
   1. ask the user to enter a *mark* in the range 0..100
   2. using an *if-else if-else* structure do the following:
      1. if the *mark* is out of range, flag an error with the erroneous value in the message
      2. otherwise:
         1. if the *mark* is 70..100 output “A”
         2. if the *mark* is 60..69 output “B”
         3. if the *mark* is 50..59 output “C”
         4. if the *mark* is 40..49 output “D”
         5. otherwise output “Fail”
   3. wrap the whole *if* statement in a *try-catch* block; thus, if the user enters “abc” for the *mark*, you will be able to cater for that. In the *catch* block, output “Exception caught!” and print the stack trace.
10. switchMathOperation()
    1. initialise two local variables: *answer* is a *double* (0.0) and *charOK* is a *boolean* (*true*).
    2. ask the user to input two *double* numbers, *num1* and *num2* (*sc.nextDouble()*).
    3. ask the user to input a character; store in a variable named *operation*.
    4. *switch* on *operation*:
       1. if it’s a ‘+’ add *num1* and *num2* together and store the sum in *answer*
       2. if it’s a ‘-’ subtract *num2* from *num1* and store the result in *answer*
       3. if it’s a ‘\*’ multiply *num1* and *num2* together and store the product in *answer*
       4. if it’s a ‘/’ divide *num1* by *num2* and store the result in *answer*
       5. any other character: output “Unknown mathematical operator “ and append the char in question to the error message and set *charOK* to *false*
    5. after the *switch*, if *charOK* is true, output the *answer*. Therefore, if we input an invalid mathematical operation, no *answer* will be output.
11. switchEnumType()
    1. firstly, just above the *public class* declare the following *enum:  
       enum Direction{* NORTH, SOUTH, EAST, WEST  
       }
    2. once that is done, do the following in the method *switchEnumType*():
       1. declare a *Direction* reference *theWay* and initialise it to *null*
       2. prompt the user to enter a string to represent the direction (N, S, E or W)
       3. convert the string entered to uppercase e.g. “n” goes to “N”
       4. switch on the string variable
          1. if its “N” set *theWay* to *Direction.NORTH* and then *break*
          2. if its “S” set *theWay* to *Direction.SOUTH* and then *break*
          3. if its “E” set *theWay* to *Direction.EAST* and then *break*
          4. if its “W” set *theWay* to *Direction.WEST* and then *break*
          5. otherwise, output the string entered following by “ is not recognised”
       5. now, switch on *theWay*
          1. if its NORTH output “Santy…” and then *break*
          2. if its SOUTH output “Penguins…” and then *break*
          3. if its EAST output “The land of the rising sun…” and then *break*
          4. if its WEST output “Hollywood…” and then *break*
       6. when testing, an invalid entry will generate an error message in the first *switch* but a *NullPointerException* in the second *switch*. Fix the error.
12. ifTemperature()
    1. declare a local variable *temperature* and initialise it to 0
    2. declare the following constants (*final int*): COLD is 0; MILD is 15; WARM is 20; VERY\_WARM is 25 and HOT is 30.
    3. ask the user to enter a temperature (an *int*)
    4. using an *if-else if-else* structure and the constants declared, code the following:
       1. if the *temperature* is<=0 output “cold”
       2. if the *temperature* isbetween 1..14 output “a little cold but ok”
       3. if the *temperature* isbetween 15..19 output “mild”
       4. if the *temperature* isbetween 20..24 output “warm”
       5. if the *temperature* isbetween 25..29 output “very warm”
       6. any *temperature* >30 output “hot”
    5. wrap the *sc.nextInt()* and the *if* statement in a *try-catch* block. In the exception handler (the *catch* block) output “Exception caught” and print the stack trace.
13. switchDaysInMonth()
    1. ask the user to enter a year and store in a local variable namely *year*
    2. ask the user to enter a month (1..12) and store in a local variable namely *month*
    3. declare a local variable *numDays* and initialise it to 0
    4. declare constants for the months of the year e.g. *final int JAN=1, FEB=2, MAR=3;*
    5. switch on month:
       1. if it’s January, March, May, July, August, October or December then set *numDays* to 31 and then *break*
       2. if its April, June, September or November then set *numDays* to 30 and then *break*
       3. if its February we need to figure out if the year is a leap year:
          1. if *year* is a multiple of 400 (use % operator) OR
          2. if *year* is a multiple of 4 AND **not** a multiple of 100
          3. then it’s a leap year i.e. set *numDays* to 29
          4. otherwise set *numDays* to 28