20/9/24

1. Procedure

declare two variables a and b

Set a to non-zero value

Set b to 0

try block:

attempt to divide a by b and store the result in another variable result

exception:

catch the ArithemeticException

display an error message

end try-catch block

end procedure

2. Array Bound of exception:

Procedure

Initialize arr[] = {5,6,7,8,9}

Set a variable index to a value greater than the array's length

try block:

Attempt to access the array element at index

exception:

Catch ArrayIndexOutOfBoundException

display error message

end try-catch block
end procedure

3.

Procedure

define NegativeValuesException extending exception

- Constructor:
  Send message to parent Class

Main

define CheckPositive(int number)

if number < 0
Throw NegativeValueException

try:
call checkPositive(number)
if no exception is thrown 'Number is valid'

catch NegativeValueException:
display the error

4.

declare
int a = 3
float b = 4.56
double c = 5.6473467
char d = 'a'
bool e = true

display the data types seperately

5. Procedure:-

Initialize Variables

Key: make, model, years

Constructor

input make, model, year

set Class Variables to input variables

Method PrintCarDetails

output: display make, model and year of the Car

Main:

Create an Object of Car with make, model and years

Call printDetails and display the details

6. Create a Matrix A with dimension 1×9

Create Matrix B with dimension 1×9

For i in range from 0 to 8

set matrix A[0][i] to i+1

For i in range from 9 to 1

Set matrix B[0][i] to i

For i from 0 to 8

print matrix A[0][i]

for i from 9 to 1

print matrix B[0][i]

7. Pseudocode:

Procedure :-

Initialize accountnumber, balance

Constructor:

input : accountnumber, initial balance

set account number and balance to input values

Method deposit:

input amount

if amount <= balance

balance = -amount

else

display insufficient balance

Main:

Create object of BankAccount with account number and initial Balance

display the balance

8. declare the Scanner class to get input

get the operator and Operand

if operator == Sum

display the Sum

else if operator == difference

return the difference between the numbers

else if Operator = *

return the product

else if operator = /

return the quotient

display the result

end procedure

9. Procedure

declare Scanner class

initialize base = 4

exponent = 3

using the inbuit function

assign pow = Mather.pow(base, exponent)

display the result

end procedure

10. Procedure:

assign n = length

expected sum = n * (n+1)/2

actual - sum = sum (numbers)

missing numbers = expected - sum - actual - sum

return missing_numbers

display the missing_numbers

end procedure