Java Exercises:

1. Write an abstract class named Arithmetic and an abstract method named add are declared . Write a class named Adder that inherits from the Arithmetic class. The Adder class must implement the add method which has two integer parameters, a and b, and calculates and returns the sum of a and b.

Your implementation of the add method must return the sum of its two parameters. The add method should print the name of the Adder class' superclass (i.e., the class that Adder inherits from).The sum of integers a and b as returned by calling the add method on an Adder object.Write a Demo class which create the instance of Adder class and call it's add method

2. Create class MyCalculator which has a single method: int power(int,int). This method should have two integer parameters, n and p, and must calculate and return np.If one or both arguments passed to it are negative, then the method must throw an exception with the following message: n and p should be non-negative.

Input Format : The parameters for your power method should be int n and int p, respectively.

Output Format: The power(int,int) method should return the result of np, or throw an exception in the event that one of the inputs was negative.

3. Write the interface called GeometricObject, which declares two abstract methods: getParameter() and getArea()Write the implementation class Circle, with a protected variable radius, which implements the interface GeometricObject.Write a test program called TestCircle to test the methods defined in Circle

4. The class ResizableCircle is defined as a subclass of the class Circle, which also implements an interface called Resizable. The interface Resizable declares an abstract method resize(), which modifies the dimension (such as radius) by the given percentage. Write the interface Resizable and the class ResizableCircle.Write a test program called TestResizableCircle to test the methods defined in ResizableCircle

5. Consider a class BankAcc with the following members:

* + Accno
  + Name
  + Address
  + Balance
  + display()
  + withdraw (long amount)
  + deposit(long amount).
* Create a subclass DepositAcc override the display () method in class DepositAcc. It only displays a message: Account type is Deposit with the statement:
* The display() method from DepositAcc is not as useful as the one of its superclass or parent class. So make the display class to perform the BankAcc’s display() function and display the message.
* Override the withdraw() method in class DepositAcc so that it first checks to see if the amount to be withdrawn is less than or equal to the balance. If it is, then it calls the withdraw() method of it’s parent class, otherwise it generates an error message.
* Create a subclass of BankAcc called CurrentAcc which will represent a current account. It will need 3 extra attributes: withdrawal limit, interest rate charged on overdrafts, number of transactions. Each time a transaction is made, the number of transactions should be increased by 1
* Write/Override methods in CurrentAcc
  + So that when a withdrawal or deposit is made, the transaction counter is incremented.
  + So that the display method displays the account type.
  + To subtract the interest due from the balance with the number of months as a parameter.
  + To withdraw money from the account, this should generate an error message if the overdraft limit is about to be exceeded.
  + To calculate the account fees owed and subtract them from the balance. The fee is 25p per transaction.

# 6. Create a class Point with the following attributes

* + x
  + y
  + Point()
  + Point(int, int)
  + getX
  + getY
  + getArea
  + translate(int dx, int dy) – moves the Point dx in the x direction and by dy in the y direction

# Create a class Circle from the Point class with members

* + radius
  + Circle()
  + Circle(int, int, int)
  + getRadius()
  + getArea()
  + grow(int)
* Create a class Square from the Point class with members
  + width
  + Square(int, int, int)
  + getWidth()
  + getArea()
  + grow(int)
* Create a class Rectangle from the Square class with members
  + height
  + Rectangle(int, int, int, int)
  + getHeight()
  + getArea()
  + grow(int,int)
* Instantiate all the classes in the main function and check for the functionalities.

7. Write a custom exception class InsufficientBalanceException. Write a class which has withDrawMoney method which takes amountToWithDraw as parameter.Inside the method, compare the balance with amountToWithdraw. If balance is less than amountToWithdraw then raise an exception InsufficientBalanceException.

8.Write a program to have try catch and finally blocks for the following exceptions

1.Divide by zero

2.Accessing array with indices

3.Call a method of using a null variable

9. Given an array of ints, return the number of 9's in the array.

Examples

arrayCount9([1, 2, 9]) → 1

arrayCount9([1, 9, 9]) → 2

arrayCount9([1, 9, 9, 3, 9]) → 3

10. Count the number of "xx" in the given string. We'll say that overlapping is allowed, so "xxx" contains 2 "xx".

Examples:

countXX("abcxx") → 1

countXX("xxx") → 2

countXX("xxxx") → 3

11. Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".

Examples:

firstHalf("WooHoo") → "Woo"

firstHalf("HelloThere") → "Hello"

firstHalf("abcdef") → "abc"

12. Given an array of ints length 3, return the sum of all the elements.

Examples:

sum3([1, 2, 3]) → 6

sum3([5, 11, 2]) → 18

sum3([7, 0, 0]) → 7

13. Given 2 ints, a and b, return their sum. However, "teen" values in the range 13..19 inclusive, are extra lucky. So if either value is a teen, just return 19.

Examples:

teenSum(3, 4) → 7

teenSum(10, 13) → 19

teenSum(13, 2) → 19

14. Given two ints, each in the range 10..99, return true if there is a digit that appears in both numbers, such as the 2 in 12 and 23. (Note: division, e.g. n/10, gives the left digit while the % "mod" n%10 gives the right digit.)

Examples

shareDigit(12, 23) → true

shareDigit(12, 43) → false

shareDigit(12, 44) → false

15. Given 3 int values, a b c, return their sum. However, if one of the values is the same as another of the values, it does not count towards the sum.

Examples:

loneSum(1, 2, 3) → 6

loneSum(3, 2, 3) → 2

loneSum(3, 3, 3) → 0

16. Modify and return the given map as follows: if the key "a" has a value, set the key "b" to have that value, and set the key "a" to have the value "". Basically "b" is a bully, taking the value and replacing it with the empty string.

Examples:

mapBully({"a": "candy", "b": "dirt"}) → {"a": "", "b": "candy"}

mapBully({"a": "candy"}) → {"a": "", "b": "candy"}

mapBully({"a": "candy", "b": "carrot", "c": "meh"}) → {"a": "", "b": "candy", "c": "meh"}