Title : Lease vs. buy car: compare new car financing options to find the optimal

Description:

The situation can be handled by encountering the root of the cause to take the cars for leasing and the issues may include is while we don’t own a car that you lease, we’re still responsible for damages. If you return the vehicle damaged at the end of the lease, we’ll have to pay fees for what the automotive dealer deems excessive wear and tear. So, it is better to buy a car rather than leasing the car or we should provide the hourly/day lease offers with low price .
It is important to tackle because, if we’re looking to get a new car, you might consider leasing it instead of buying it outright. While car leases typically come with lower monthly payments, you won’t actually own the car. Buying a car, on the other hand, means you’ll be purchasing an asset, which can be worth making higher payments.

We are planning to develop a website to compare new car financing options to find the optimal in Leasing and buying a car. By developing this website, it will be easy for people to lease or buy a car. Due to severe market competition, people are opting to lease a car rather than buying a car. But, if you’re looking to get a new car, you might consider leasing it instead of buying it outright.On one hand, buying involves higher monthly costs, but you own an asset your vehicle in the end. On the other hand, a lease has lower monthly payments and lets you drive a vehicle that may be more expensive than you could afford to buy, but you get into a cycle in which you never stop paying for the vehicle.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import java.lang.Object;
import java.util.Arrays;
import java.io.*;
import java.util.*;
import java.lang.Math;

/**
 *
 * @author sidharth
 */
public class AlgorithmsHW3 {

    /**
     *
******************************************************************************
******************
     *
     * PROBLEM SET 1
     * [10 points] fibonacci_exponential: compute nth fibonacci number with
an exponential running time
     * [10 points] fibonacci_linear: compute nth fibonacci number with an
exponential running time
     * [20 points] fibonacci_log: compute nth fibonacci number with a
logarithmic running time
     *
     * [10 points] Plot a graph showing the timings to compute the first 30
fibonacci numbers using all three methods. And for the first 45 fibonacci
numbers using the linear and logarithmic method.
                 X axis should be for the fibonacci number and y axis
should be for time.
*****************************************************************************
*******************
     */
    public int fibonacci_exponential(int n) {

      if(n==0){
        return 0;
     }else if(n== -1){
     return -1;
         }
         else{
        return fibonacci_exponential(n-1) + fibonacci_exponential(n-2);
          }

    }

    public int fibonacci_linear(int n) {

     //array declaration for storing fibonacci numbers
  int a[] = new int[n + 1];
  int i;
  a[0] = 0;
  if(n > 0){
    a[1] =1;
    for(i = 2; i <= n; i++){
      a[i] = a[i - 1] + a[i - 2];
    }
  }
        return a[n];
    }
    }

    public int fibonacci_log(int n) {

     // TODO: Implement this
        return -1;
    }





    /**
     *
*****************************************************************************
******************
     *
     * PROBLEM SET 2
     * [20 points] You are climbing a staircase. It takes n steps to reach
the top. Each time you can either climb 1 or 2 steps. In how many distinct
ways can you climb to the top?
     * Example Input: n = 3  Output: 3 | Explanation: (1 step + 1 step + 1
step), (1 step + 2 steps), and (2 steps + 1 step)
     *
     * [5 points] Print out the time take to find solution for n=0 to n=45

*****************************************************************************
```

```
*******************
     */
    int climbStairs(int n) {

      // TODO: Implement this
        return -1;
    }


    /**
     *
******************************************************************************
******************
     *
     * PROBLEM SET 3
     *
     * [20 points] Given a triangle array, return the minimum path sum from
top to bottom.
     * For each step, you may move to an adjacent number of the row below (if
you are on index i on the current row, you may move to either index i or
index i + 1 on the next row).
     * Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
       Output: 11
       Explanation: The triangle looks like:
              2
             3 4
            6 5 7
           4 1 8 3
     The minimum path sum from top to bottom is 2 + 3 + 5 + 1 = 11.

     [5 points]
     Print out the triangle (only for triangle with 4 levels) and the answer
     Print out the correct answer for all triangles (from level 1 to 40)

******************************************************************************
*******************
     */
    public int minimumTotal(List<List<Integer>> triangle)
    {
      for (int i=0; i < triangle.size(); i++)
      {
        List<Integer> tlist = triangle.get(i);
        for (int j=0; j < tlist.size(); j++)
        {
          System.out.print(tlist.get(j)+ " ");
        }
        System.out.println();
      }
      System.out.println();


      // TODO: Implement this
        return -1;
    }
```