```
/*
 * A one-way street [0, inf) of infinite length has a bus stop every mile. A
   passenger is charged according to the number of miles he rides on the bus.
The bus cannot travel
   more than m miles (m ≥ 1) each time (in other words, the bus can travel 1
mile, 2 miles,..., m
   miles non-stop. In case a rider wants to travel more than m miles, the trip
has to be taken by
   several bus rides. For example, a trip of m + 3 miles can be taken as bus
rides of 1 miles, 3 miles,
   and m − 1 miles; or m miles, followed by 3 miles, etc. Similarly, a trip
of m − 1 miles could be a
   ride of m − 1 miles, or a ride of 2 miles followed by m − 3 miles, or even
m − 1 rides of 1 mile each.
 */
package cosc3p03_assign3;

/**
 *
 * @author pw12nb
 */
public class Question_3 {
    int m;
    int prices[];
    int ks[];
    int distances[];

    public Question_3(int[] prices)
    {
        this.m = prices.length;
        this.prices = prices;
    }

    public Question_3(int m)
    {
        this.m = m;
        prices = new int[m];
        initializePrices(prices);
    }

    private void initializePrices(int[] prices) {
```

```
        prices[0] = 0;
        for(int i = 1; i < m; i++)
        {

            prices[i] = (int)(Math.random()*50);
        }
    }

    private void initializeDistance(int[] distance, int n) {
        distance[0] = 0;
        for(int i = 1; i <= n; i++)
        {
            distance[i] = Integer.MAX_VALUE;
        }
    }

    public int getCheapestPath(int n)
   {
       ks = new int[n+1];
       this.distances = new int[n+1];
       initializeDistance(distances, n);
       for(int i = 1; i <= n; i++)
       {
           //if the indeces are the same
            int min = Integer.MAX_VALUE;
            int k = 0;

           for(int j = 0; j<i; j++)
           {
               if(distances[j] == Integer.MAX_VALUE || distances[i-j] ==
Integer.MAX_VALUE)
               {
                   if((j==i || i-j==i) && i < m)
                   {
                       distances[i] = prices[i];
                   }
                   else
                       continue;
               }

                int val = (distances[j] + distances[i-j]);
```

```
                if(val < min){
                    k = j;
                    min = val;
                }
            }
            ks[i] = k;
            distances[i] = min;
            System.out.println("the k for 0 to "+k+" to "+i+": with cost of
"+min);
        }
        return distances[n];
    }
}
```