```java
/**
 * Nombres entiers et boucles.
 *
 * @author      gamca174 (Gamboa Carlos) / olial319 (Olinger Alex)
 * @version     10/03/2016 07:13:16
 * Classe:      11TG
 */
public class SimpleCalculationWithTwoInts
{
     private int a;
     private int b;

     public SimpleCalculationWithTwoInts(int pA, int pB)
     {
          a = Math.abs(pA);
          b = Math.abs(pB);
     }

     public void printAll()
     {
          if (a > b)
          {
               swap();
          }

          int i = a;
          while (i <= b)
          {
               System.out.println(i);
               i++;
          }
     }

     public int calculateSumEven()
     {
          if (a > b)
          {
               swap();
          }

          int sum, i;
          sum = 0;

          i = a;
          while (i <= b)
          {
               if (i % 2 == 0)
               {
                    sum = sum + i;
               }
               i++;
          }

          return sum;
     }

     public double calculatePower()
     {
          double res = 1;
          int i;

          // pas de swap car a exp b est différent de b exp a

          i = 1;
          while (i <= b)
          {
               res = res * a;
               i++;
          }

          return res;
     }

     public void swap()
     {
          int tmp;

          tmp = a;
          a   = b;
          b   = tmp;
     }
```

```
    public int GCD_search()
    {
        return computeGCDSearch(a, b);
    }

    public int GCD_Euclid()
    {
        return computeGCDEuclid(a, b);
    }

    public int GCD_OptimizedEuclid()
    {
        return computeGCDOptimizedEuclid(a, b);
    }

    public int LCM_search()
    {
        return computeLCMSearch(a, b);
    }

    public int LCM_shortcut()
    {
        return computeLCMShortcut(a, b);
    }

    public int computeGCDSearch(int pA, int pB)
    {
        int gcd = Math.min(pA, pB);
        if (gcd == 0)
            return Math.max(pA, pB);

        while ((pA%gcd != 0) || (pB%gcd != 0))
        {
            gcd--;
        }
        return gcd;
    }

    public int computeGCDEuclid(int pA, int pB)
    {
        // Cas exceptionnel: on modifie les paramètres

        if (pA == 0)
        {
            return pB;
        }
        else
        {
            while (pB != 0)
            {
                if (pA > pB)
                    pA = pA - pB;
                else
                    pB = pB - pA;
            }
        }
        return pA;
    }

    public int computeGCDOptimizedEuclid(int pA, int pB)
    {
        // Cas exceptionnel: on modifie les paramètres

        int h;
        while (pB != 0)
        {
            h = pA % pB;
            pA = pB;
            pB = h;
        }
        return pA;
    }

    public int computeLCMSearch(int pA, int pB)
    {
        if ((pA == 0) || (pB == 0))
            return 0;

        int lcm = Math.max(pA, pB);
        while ((lcm % pA != 0) || (lcm % pB != 0))
        {
            lcm = lcm + 1;
```

```
        }
        return lcm;
    }

    public int computeLCMShortcut(int pA, int pB)
    {
        if ((pA == 0) || (pB == 0))
            return 0;

        return (pA*pB) / computeGCDSearch(pA, pB);
    }
}
```