```java
/**
 * Méthodes standard sur des nombres entiers.
 *
 * @author     gamca174 (Gamboa Carlos) / olial319 (Olinger Alex)
 * @version    14/04/2016 07:13:16
 * Classe:     11TG
 */
public class IntegerNumber
{
     private int n;

     public IntegerNumber(int pN)
     {
          n = Math.abs(pN);
     }

     public int getNumber()
     {
          return n;
     }

     public boolean isEven()
     {
          // version 1
          if (n%2 == 0)
               return true;
          else
               return false;

          // version 2 (mieux)
          // return n%2==0;
     }

     public boolean isPrime()
     {
          return isPrimeOptimized();
     }

     public boolean isPrimeClassic()
     {
          // Version classique, non-optimisée - plus facile à comprendre
          int count = 0;
          for (int i=1; i<=n; i++)
          {
               if (n%i == 0)
                    count++;
          }

          if (count == 2)
               return true;
          else
               return false;

          // ou encore mieux:
          //   return (count == 2);
     }

     public boolean isPrimeOptimized()
     {
          // Version optimisée
          if (n == 1)
               return false;
          if (n == 2)
               return true;

          int i = 2;
          boolean found = false;
          int limit = ((int)Math.sqrt(n))+1;
          // Remarque:
          //   le "(int)" et le "+1" permettent d'éviter les problèmes d'arrondis et de précision
          //   en théorie ils sont de trop et avec précision parfaite (qui n'existe pas en JAVA) il faut écrire:
          //      double limit = Math.sqrt(n);

          while (!found && (i<=limit))
          {
               if (n%i == 0)
                    found = true;  // un diviseur de n trouvé!
               i++;
          }

          return !found;
     }
```

```
    public int sumOfDividers(int pN)
    {
        // cette méthode est utilisée plusieurs fois...
        int sumDiv=0;
        for (int i=1; i<=pN; i++)
        {
            if (pN%i == 0)
                sumDiv = sumDiv + i;
        }
        return sumDiv;
    }

    public int sumOfDividers()
    {
        return sumOfDividers(n);
    }

    public boolean isPerfect()
    {
        return sumOfDividers() == 2*n;
    }

    public boolean isDeficient()
    {
        return sumOfDividers() < 2*n;
    }

    public boolean isAbundant()
    {
        return sumOfDividers() > 2*n;
    }

    public boolean isFriendlyTo(int pN)
    {
        return sumOfDividers() == sumOfDividers(pN);
    }

    public int reverse(int pN)
    {
        int res = 0;
        int val = pN;
        while (val!=0)
        {
            res = res*10 + val%10;
            val = val/10;
        }
        return res;
    }

    public int reverse()
    {
        return reverse(n);
    }

    public boolean isPalindrome()
    {
        int rev = reverse();
        return n==rev;
    }
}
```

```java
/**
 * Teste les méthodes / algorithmes programmés dans IntegerNumber.
 *
 * @author     gamca174 (Gamboa Carlos) / olial319 (Olinger Alex)
 * @version    14/04/2016 07:13:16
 * Classe:     11TG
 */
public class TestIntegerNumber
{
     /**
      * Programme principal.
      */
     public static void main(String[] args)
     {
          IntegerNumber a;

          System.out.println("*** nombres pairs ou impairs ***");
          a = new IntegerNumber(1); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(2); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(3); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(4); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(5); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(6); System.out.println("  n="+a.getNumber()+"   --> isEven() = "+a.isEven());
          a = new IntegerNumber(15); System.out.println("  n="+a.getNumber()+"  --> isEven() = "+a.isEven());
          a = new IntegerNumber(16); System.out.println("  n="+a.getNumber()+"  --> isEven() = "+a.isEven());
          a = new IntegerNumber(17); System.out.println("  n="+a.getNumber()+"  --> isEven() = "+a.isEven());
          a = new IntegerNumber(121); System.out.println("  n="+a.getNumber()+" --> isEven() = "+a.isEven());
          a = new IntegerNumber(122); System.out.println("  n="+a.getNumber()+" --> isEven() = "+a.isEven());
          a = new IntegerNumber(123); System.out.println("  n="+a.getNumber()+" --> isEven() = "+a.isEven());

          System.out.println();
          System.out.println("*** nombres premiers ***");
          a = new IntegerNumber(1); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(2); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                "  / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(3); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                "  / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(4); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(5); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                "  / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(6); System.out.println("  n="+a.getNumber()+"    --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(15); System.out.println("  n="+a.getNumber()+"   --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(16); System.out.println("  n="+a.getNumber()+"   --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(17); System.out.println("  n="+a.getNumber()+"   --> isPrimeClassic() = "+a.isPrimeClassic()+
                "  / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(121); System.out.println("  n="+a.getNumber()+"  --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(122); System.out.println("  n="+a.getNumber()+"  --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());
          a = new IntegerNumber(123); System.out.println("  n="+a.getNumber()+"  --> isPrimeClassic() = "+a.isPrimeClassic()+
                " / isPrimeOptimized() = "+a.isPrimeOptimized());

          System.out.println();
          System.out.println("*** calculs sur la somme des diviseurs ***");
          a = new IntegerNumber(1); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "     / 2*n = "+(a.getNumber()*2)+"     / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                "  / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(2); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "     / 2*n = "+(a.getNumber()*2)+"     / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                "  / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(3); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "     / 2*n = "+(a.getNumber()*2)+"     / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                "  / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(4); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "     / 2*n = "+(a.getNumber()*2)+"     / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                "  / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(5); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "     / 2*n = "+(a.getNumber()*2)+"    / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                "  / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(6); System.out.println("  n="+a.getNumber()+"    / sumOfDividers() = "+a.sumOfDividers()+
                "    / 2*n = "+(a.getNumber()*2)+"    / isPerfect() = "+a.isPerfect()+"  / isDeficient() = "+a.isDeficient()+
                " / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(12); System.out.println("  n="+a.getNumber()+"   / sumOfDividers() = "+a.sumOfDividers()+
                "    / 2*n = "+(a.getNumber()*2)+"    / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
                " / isAbundant() = "+a.isAbundant());
          a = new IntegerNumber(28); System.out.println("  n="+a.getNumber()+"   / sumOfDividers() = "+a.sumOfDividers()
                "    / 2*n = "+(a.getNumber()*2)+"    / isPerfect() = "+a.isPerfect()+"  / isDeficient() = "+a.isDeficient()+
```

```
            " / isAbundant() = "+a.isAbundant());
        a = new IntegerNumber(496); System.out.println("  n="+a.getNumber()+"  / sumOfDividers() = "+a.sumOfDividers()+
            "   / 2*n = "+(a.getNumber()*2)+"   / isPerfect() = "+a.isPerfect()+"  / isDeficient() = "+a.isDeficient()+
            " / isAbundant() = "+a.isAbundant());
        a = new IntegerNumber(8128); System.out.println("  n="+a.getNumber()+" / sumOfDividers() = "+a.sumOfDividers()+
            " / 2*n = "+(a.getNumber()*2)+" / isPerfect() = "+a.isPerfect()+"  / isDeficient() = "+a.isDeficient()+
            " / isAbundant() = "+a.isAbundant());
        a = new IntegerNumber(8129); System.out.println("  n="+a.getNumber()+" / sumOfDividers() = "+a.sumOfDividers()+
            "  / 2*n = "+(a.getNumber()*2)+" / isPerfect() = "+a.isPerfect()+" / isDeficient() = "+a.isDeficient()+
            "  / isAbundant() = "+a.isAbundant());

        System.out.println();
        System.out.println("*** renversement de valeurs ***");
        a = new IntegerNumber(1); System.out.println("  n="+a.getNumber()+"     / reverse() = "+a.reverse()+
            "     / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(2); System.out.println("  n="+a.getNumber()+"     / reverse() = "+a.reverse()+
            "     / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(9); System.out.println("  n="+a.getNumber()+"     / reverse() = "+a.reverse()+
            "     / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(10); System.out.println("  n="+a.getNumber()+"    / reverse() = "+a.reverse()+
            "     / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(11); System.out.println("  n="+a.getNumber()+"    / reverse() = "+a.reverse()+
            "    / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(19); System.out.println("  n="+a.getNumber()+"    / reverse() = "+a.reverse()+
            "    / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(1234); System.out.println("  n="+a.getNumber()+"  / reverse() = "+a.reverse()+
            "  / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(4334); System.out.println("  n="+a.getNumber()+"  / reverse() = "+a.reverse()+
            "  / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(3434); System.out.println("  n="+a.getNumber()+"  / reverse() = "+a.reverse()+
            "  / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(12321); System.out.println("  n="+a.getNumber()+" / reverse() = "+a.reverse()+
            " / isPalindrome() = "+a.isPalindrome());
        a = new IntegerNumber(12312); System.out.println("  n="+a.getNumber()+" / reverse() = "+a.reverse()+
            " / isPalindrome() = "+a.isPalindrome());
    }
}
```