Modélisation Objet Java 7

http://www.centraliup.fr.st

1. Entrées-Sorties

Ecrire le programme Java suivant :

- lire au clavier deux nombres *a* et *b*,
- calculer leur somme *s*, leur produit *p* et les afficher,
- intervertir les valeurs de *s* et de *p* et les afficher.

```
import java.io.*;

class tp1ex1
{
    public static String saisirLigne(String s) throws IOException
    {
        BufferedReader inr =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print(s);
        String line = inr.readLine();
        return line;
    }

    public static void main(String arguments[]) throws IOException
    {
        // Lecture de a et b :
        int a,b;
        String chaine;
        chaine = saisirLigne("Entrez un premier entier : ");
        a=Integer.parseInt(chaine);
        chaine = saisirLigne("Entrez un second entier : ");
        b=Integer.parseInt(chaine);

        // Somme et produit :
        int s=a+b;
        int p=a*b;
        System.out.print("Somme : ");
        System.out.println(s);
        System.out.print("Produit : ");
        System.out.println(p);

        // Echange des deux valeurs :
        int t=p; p=s; s=t;
        System.out.print("Valeur de s : ");
        System.out.println(s);
        System.out.print("Valeur de p : ");
        System.out.println(p);

        // Fin :
        System.exit(0);
    }
};
```

2. Itération

Ecrire un programme Java qui :

- calcule la somme des N premiers entiers
- calcule la somme des carrés des N premiers entiers
- calcule le produit des N premiers entiers

```
import java.io.*;

class tp1ex2
{
    public static String saisirLigne(String s) throws IOException
    {
        BufferedReader inr =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print(s);
        String line = inr.readLine();
        return line;
    }

    public static void main(String args[]) throws IOException
    {
        // Demande d'un entier :
        int n, s1=0, s2=0, p=1;
        String chaine;
        chaine = saisirLigne("Entrez un entier positif : ");
        n = Integer.parseInt(chaine);

        // Calcul de la somme des n premiers entiers :
        for(int i=1; i<=n; s1+=i, i++);

        // Calcul de la somme des n premiers entiers au carre :
        for(int i=1; i<=n; s2+=i*i, i++);

        // Calcul du produit des n premiers entiers :
        for(int i=1; i<=n; p*=i, i++);

        // Affichage des resultats :
        System.out.print("Somme des entiers : ");
        System.out.println(s1);
        System.out.print("Somme des carres des entiers : ");
        System.out.println(s2);
        System.out.print("Produit des entiers : ");
        System.out.println(p);

        System.exit(0);
    }
};
```

3. Puissance

Ecrire une fonction permettant de calculer X à la puissance N, pour X quelconque et N entier, en utilisant le développement de N en base 2.

```
import java.io.*;

class tp1ex3
{
    public static String saisirLigne(String s) throws IOException
    {
        BufferedReader inr =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print(s);
        String line = inr.readLine();
        return line;
    }

    public static int puissance(float val, int puiss)
    {
        if(puiss>0)
        {
            if(puiss%2==1) return val*puissance(val,puiss-1);
            else
            {
                int p=puissance(val, puiss/2);
                return p*p;
            }
        }
        return 1;
    }

    public static void main(String args[]) throws IOException
    {
        // Demande de x et n :
        int n, res=1;
        float x;
        String chaine;
        chaine = saisirLigne("Entrez votre X : ");
        x = Float.parseFloat(chaine);
        chaine = saisirLigne("CollÉgue, entre ton N : ");
        n = Integer.parseInt(chaine);

        // Calcul par decomposition de n en base 2
        res = puissance(x,n);

        // Resultat et fin :
        System.out.print("Resultat de ce calcul intense : ");
        System.out.println(res);
        System.exit(0);
    }
};
```

4. Tableaux

Etant donné un tableau *T* de *N* nombres positifs ou nuls, on se propose d'écrire un programme Java qui enlève les éléments nuls et qui récupère la place. Plus précisément, si *T*[*i*] = 0, alors les éléments *T*[*i*+1], *T*[*i*+2], ? sont décalés à gauche.

```
import java.io.*;

class tp1ex4
{
    public static int reduction(int tab[], int nb)
    {
        int nbZeros=0,pos=0;
        for(int i=0; i<nb; i++)
        {
            if(tab[pos]==0)
            {
                // Decale les valeurs a partir de pos
                for(int j=pos; j<nb-1-nbZeros; j++)
                    tab[j]=tab[j+1];
                 // Fixe le dernier element a 0 :
                 tab[nb-1]=0;
                 nbZeros++;
            }
            else pos++;
        }
        return nbZeros;
    }

    public static void main(String args[])
    {
        // Initialisation :
        int N=12;
        int tab[]={0,1,3,0,2,0,0,1,5,0,9,0};

        // Affichage du tableau initial :
        System.out.println("Tableau initial :");
        for(int i=0; i<N; i++)
        {
            System.out.print(tab[i]);
            System.out.print(" ");
        }

        // Recuperation de l'espace laisse par les zeros :
        int nZeros=reduction(tab,N);
        int tabred[]=new int[N-nZeros];

        // Affichage du tableau final et copie :
        System.out.println("\nTableau reduit :");
        for(int i=0; i<N-nZeros; i++)
        {
            tabred[i]=tab[i];
            System.out.print(tabred[i]);
            System.out.print(" ");
        }
        // Provoque la destruction du premier tableau :
        tab=tabred;
        System.out.println("\nGain de place : "+nZeros+" entiers.");

        // Fin :
        System.exit(0);
    }
};
```

5. Tableaux triés

Programmer en Java le tri par insertion dont le principe est le suivant :
si les valeurs x_1 ... x_{k-1} sont ordonnées,

- chercher le rang j de x_k parmi x_{k-1}, x_{k-2}, ...
- pousser ces éléments vers la droite
- ranger x_k à la place j

Les valeurs x_1 ... x_k seront alors ordonnées.
Il suffit d'appliquer ce principe n-1 fois en partant de la seule valeur x_1.

```
import java.io.*;

class tp1ex5
{
    public static void tri(int tab[], int nb)
    {
        int j;
        int tmp;
        for(int i=1; i<nb; i++)
        {
            j=i-1;
            // Sauvegarde de l'element :
            tmp=tab[i];
            // Recherche de l'endroit a inserer :
            while(j>=0 && tab[j]>tab[i]) j--;
            // Decalage vers la droite de j a i :
            for(int k=i-1; k>=j+1; k--)
                tab[k+1]=tab[k];
            // Puis placement de la valeur a sa place :
            tab[j+1]=tmp;

            System.out.print("Tour #");
            System.out.print(i);
            System.out.print(" --> ");
            for(int h=0; h<nb; h++)
            {
                System.out.print(tab[h]);
                System.out.print(" ");
            }

            System.out.println("");
        }
    }

    public static void main(String args[])
    {
        // Initialisation :
        int N=12;
        int tab[]={7,1,5,0,4,0,9,6,8,5,2,3};

        // Affichage du tableau initial :
        System.out.println("Tableau initial :");
        for(int i=0; i<N; i++)
        {
            System.out.print(tab[i]);
            System.out.print(" ");
        }
        System.out.println("\n");

        // Tri par insertion :
        tri(tab,N);

        // Affichage du tableau final :
        System.out.println("\nTableau trie :");
        for(int i=0; i<N; i++)
        {
            System.out.print(tab[i]);
            System.out.print(" ");
        }

        // Fin :
        System.out.println("");
        System.exit(0);
    }
};
```

6. Tableaux à deux dimensions

Ecrire une fonction calculant le produit de 2 matrices.

```
import java.io.*;

class tp1ex6
{
    public static void afficheMatrice(String titre, int M[][],
    int nl, int nc)
    {
        System.out.println(titre);
        System.out.println("---------");
        for(int i=0; i<nl; i++)
        {
            for(int j=0; j<nc; j++)
                System.out.print(M[i][j]+"\t");
            System.out.print("\n");
        }
        System.out.println("");
    }

        public static void main(String args[])
        {
            // Initialisation :
            int NLIG_A=3;
            int NCOL_A=4;
            int NLIG_B=4;
            int NCOL_B=5;
            int A[][]={{2,7,-1,0},{3,-2,1,-5},{-1,3,0,0}};
            int B[][]={{7,-1,3,0,3},{-1,-1,-1,-1,-1},{3,-3,6,0,1},
                 {5,-2,-4,2,1}};
            int C[][]=new int[NLIG_A][NCOL_B];

            // Affichage des matrices A & B :
            afficheMatrice("Matrice A : ", A, NLIG_A, NCOL_A);
            afficheMatrice("Matrice B : ", B, NLIG_B, NCOL_B);

            // Produit matriciel :
            for(int i=0; i<NLIG_A; i++)
                for(int j=0; j<NCOL_B; j++)
                    for(int k=0; k<NCOL_A; k++)
                        C[i][j]+=A[i][k]*B[k][j];

            // Affichage de la matrice C :
            afficheMatrice("Matrice C : ", C, NLIG_A, NCOL_B);

            System.out.println("");
            System.exit(0);
        }
};
```

7. Récursivité

Ecrire un programme <u>récursif</u> qui détermine si une chaîne contenue dans un tableau de caractères est un palindrome.

```
import java.io.*;

class tp1ex7
{
    public static String saisirLigne(String s) throws IOException
    {
        BufferedReader inr =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print(s);
        String line = inr.readLine(); return line;
    }

    public static boolean estPal(String chaine, int deb, int fin)
    {
        if(fin-deb<=1) return true;
        return chaine.charAt(deb)==chaine.charAt(fin) &&
             estPal(chaine,deb+1,fin-1);
    }

    public static boolean estPalindrome(String chaine)
    {
        return estPal(chaine,0,chaine.length()-1);
    }

    public static void main(String args[]) throws IOException
    {
        // Initialisation :
        String chaine=saisirLigne("Entrez une chaine : ");

        // Test de pater.. euh palindrome :
        if(estPalindrome(chaine))
            System.out.println("Bravo, c'est un palindrome !");
        else System.out.println("Perdu ! Retentez votre chance...");

        System.out.println("");
        System.exit(0);
    }
};
```

8. Fusion & récursivité

Programmer en Java le tri par fusion <u>récursif</u> dont le principe est le suivant :

- trier chaque moitié du tableau
- les fusionner de manière à obtenir un tableau ordonné

```
import java.io.*;

class tp1ex8
{
    public static void fusion(int tab[], int deb, int mil, int fin)
    {
        // Initialisation des bornes :
        int deb1=deb, fin1=mil, deb2=mil+1, fin2=fin;
        // Index de parcours des deux parties :
        int index1=deb1, index2=deb2;
        // Tableau de stockage tamporaire des valeurs :
        int tampon[]=new int[fin-deb+1];

        // Boucle principale :
        for(int i=0; i<fin-deb+1; i++)
        {
            if(index2==fin2+1) tampon[i]=tab[index1++];
            else if(index1==fin1+1) tampon[i]=tab[index2++];
                 else if(tab[index1]<tab[index2])
                     tampon[i]=tab[index1++];
                     else tampon[i]=tab[index2++];
        }

        // Mise a jour du tableau :
        for(int i=0; i<fin-deb+1; i++)
        {
            tab[i+deb]=tampon[i];
        }
    }

    public static void trifusion(int tab[], int deb, int fin)
    {
        if(fin-deb>=1)
        {
            int mil=(deb+fin)/2;
            trifusion(tab, deb, mil);
            trifusion(tab, mil+1, fin);
            fusion(tab, deb, mil, fin);
        }
    }

    public static void tri(int tab[], int nb)
    {
        trifusion(tab, 0, nb-1);
    }

    public static void main(String args[])
    {
        // Initialisation :
        int N=15;
        int tab[]={7,1,5,1,4,1,1,9,7,6,8,4,5,2,3};

        // Affichage du tableau initial :
        System.out.println("Tableau initial :");
        for(int i=0; i<N; i++)
        {
            System.out.print(tab[i]);
            System.out.print(" ");
        }

        // Tri par fusion :
        tri(tab,N);

        // Affichage du tableau final :
        System.out.println("\nTableau trie :");
        for(int i=0; i<N; i++)
        {
            System.out.print(tab[i]);
            System.out.print(" ");
        }

        System.out.println("");
        System.exit(0);
    }
};
```

10. Première classe

Créer une classe Entiers permettant de manipuler des ensembles ordonnés sans répétition de nombres entiers. Pour cela :

- on rangera les différents éléments de l'ensemble dans un tableau
- on rangera la taille de ce tableau ainsi que le nombre courant d'éléments dans deux autres variables membres
- on définira un constructeur avec un argument qui précisera la taille du tableau
- on définira une fonction membre " cardinal " sans argument qui renvoie le nombre d'éléments de l'ensemble
- on définira une fonction membre " appartient " à un argument entier qui renvoie *true* si cet élément appartient à l'ensemble, *false* sinon
- on définira une fonction membre " insérer " à un argument entier qui insère cet élément à sa place (s'il n'appartient pas déjà à l'ensemble)
- on définira une fonction membre " copier " sans argument qui renvoie une copie de l'ensemble
- on définira une fonction membre " supprimer " à un argument entier qui supprime cet élément de l'ensemble (s'il s'y trouve)
- on définira une fonction membre " union " à un argument " Entiers " qui renvoie l'union de l'ensemble de référence et de cet argument

```
import java.io.*;

class Entiers
{
    private int tab[];
    private int taille;
    private int nbElem;

    public Entiers(int t)
    {
        tab=new int[t];
        nbElem=0;
        taille=t;
    }

    public int cardinal() {return nbElem;}

    public boolean appartient(int val)
    {
        int i=0;
        // Le tableau est ordonne, tirons-en parti :
        while(i<nbElem && tab[i]<val) i++;
        if(i<nbElem && tab[i]==val) return true; else return false;
    }

    public boolean inserer(int val)
    {
        // Verifie qu'il reste de la place :
        if(nbElem==taille-1) return false;

        // Puis recherche l'emplacement pour inserer l'element :
        int pos=0;
        while(pos<nbElem && tab[pos]<val) pos++;

        // Verifie que l'element n'existe pas deja :
        if(pos==nbElem || tab[pos]>val)
        {
            // Alors decale les elements suivants :
            for(int i=nbElem+1; i>pos; i--) tab[i]=tab[i-1];

            // Puis place l'element :
            tab[pos]=val;
            nbElem++;
            return true;
        }
        else return false;
    }

    public Entiers copier()
    {
        // Alloue de la memoire pour le nouvel ensemble :
        Entiers e = new Entiers(taille);
        e.nbElem = nbElem;
        // Copie les valeurs une a une...
        for(int i=0; i<nbElem; i++)
            e.tab[i]=tab[i];
        return e;
    }

    public void supprimer(int val)
    {
        // Verifie que le tableau n'est pas vide :
        if(nbElem==0) return;

        // Recherche l'element a supprimer :
        int i=0;
        while(i<nbElem && tab[i]<val) i++;

        // Element absent :
        if(i>=nbElem || tab[i]>val) return;

        // Suppression : decalage des elements suivants :
        for(i=i; i<nbElem-1; i++)
            tab[i]=tab[i+1];
        nbElem--;
    }

    public void print() throws IOException
    {
        // Affiche des infos sur l'ensemble :
        System.out.println("Taille : "+taille);
        System.out.println("Nombre d'elements : "+nbElem);
        System.out.print("Ensemble : ");
        // Puis affiche les valeurs :
        for(int i=0; i<nbElem; i++) System.out.print(tab[i] + ";");
        System.out.print("\n");
    }

    public Entiers union(Entiers e)
    {
        Entiers nouv=new Entiers(taille + e.taille);
        int pointeur1=0, // Element en cours de this
            pointeur2=0; // Element en cours de e

        int doublons=0;  // Compteur de doublons

        // Boucle de fusion des deux tableaux ordonnes
        // (en fait la meme que celle du tri a fusion) :
        for(int i=0; i<nbElem+e.nbElem; i++)
        {
            if(pointeur2==e.nbElem) nouv.tab[i]=tab[pointeur1++];
            else if(pointeur1==nbElem) nouv.tab[i]=e.tab[pointeur2++];
            else if(tab[pointeur1]<e.tab[pointeur2]) nouv.tab[i]=tab[pointeur1++];
            else if(tab[pointeur1]>e.tab[pointeur2])nouv.tab[i]=e.tab[pointeur2++];
            else {doublons++;nouv.tab[i]=tab[pointeur1++];pointeur2++;}
            // Arrive ci-dessus, les elements sont egaux !
        }

        // Gestion du nombre d'elements du nouvel ensemble :
        nouv.nbElem = nbElem + e.nbElem - doublons;

        // On aurait pu utiliser la fonction inserer(int), qui gere
        // les doublons, mais on aurait fait d'innombrables parcours
        // de tableau pour rien, alors qu'ici on ne fait qu'un seul
        // parcours des trois tableaux a la fois.
        return nouv;
    }
};


class tp1ex10
{
    // Programme de test de la classe :
    public static void main(String arguments[]) throws IOException
    {
        // Cree deux ensembles d'entiers de taille 100 :
        Entiers ensemble1, ensemble2, ensemble3, ensemble4;
        ensemble1 = new Entiers(100);

        // Manipulation sur les ensembles :
        // (insertion, suppression, copie)
        ensemble1.inserer(3); ensemble1.inserer(8);
        ensemble1.inserer(2); ensemble1.inserer(5);
        ensemble3=ensemble1.copier();

        ensemble1.inserer(15); ensemble1.inserer(7);
        ensemble1.inserer(12); ensemble1.inserer(1);
        ensemble2=ensemble1.copier();

        ensemble2.supprimer(8); ensemble2.supprimer(5);
        ensemble2.supprimer(1); ensemble2.supprimer(8);
        ensemble2.supprimer(7); ensemble2.supprimer(15);
        ensemble2.supprimer(1);

        ensemble2.inserer(4); ensemble2.inserer(9);
        ensemble2.inserer(11); ensemble2.inserer(6);

        // Tests d'appartenance :
        if(ensemble1.appartient(5))
            System.out.println("5 est dans l'ensemble 1 !");
        else System.out.println("5 n'est pas dans l'ensemble 1 !");
        if(ensemble1.appartient(9))
            System.out.println("9 est dans l'ensemble 1 !");
        else System.out.println("9 n'est pas dans l'ensemble 1 !");

        // Affichage :
        System.out.println("\nENSEMBLE #1 -----------------------");
        ensemble1.print();
        System.out.println("\nENSEMBLE #2 -----------------------");
        ensemble2.print();
        System.out.println("\nENSEMBLE #3 -----------------------");
        ensemble3.print();

        // Union :
        ensemble4=ensemble1.union(ensemble2);
        System.out.println("\nUNION DES ENSEMBLES #1 et #2 ------");
        ensemble4.print();

        System.out.println("");
        System.exit(0);
    }
};
```