Exercice 1 : (6 points)

Quels résultats fournit le programme suivant ?

```
class A {
    public void f(double a) { System.out.println ("A==>f(double "+ a +" )"); }
    public void g() { System.out.println ("A==>g()"); }
}
class B extends A {
    public void f(int q)   { System.out.println ("B==>f(int="   + q + ") ") ; }
}
class C extends B {
    public void f(double q) { System.out.println ("C==>f(double="   + q + ") "); }
    public void f(char c)   { System.out.println ("C==>f(char='"   + c + "') ") ;}
}
class D extends C {
    public void g() { System.out.println ("D==>g()");}
}
class E extends C{
    public void f(char x)  { System.out.println ("E==>f(char='"  + x + "') ") ; }
    public void f(double t)    { System.out.println ("E==>f(double="+ t +") ") ; }
    public void g(char car) { System.out.println ("E==>g(char='"+ car +"')");}
}
class F extends E{
    public void f(double t)    { System.out.println ("F==>f(double="+ t +") ") ; }
    public void g() { System.out.println ("F==>g()");}
}
public class Surdef
{ public static void main (String arg[])
  { char car1='A',car2='Z' ;
    double x=4,y=6;
    A a = new A() ; System.out.println("Objet a:"); a.f(x) ; a.g() ;
    B b = new B() ; System.out.println("Objet b:"); b.f(x) ; b.g() ;
    C c = new C() ; System.out.println("Objet c:"); c.f(car1) ; c.f(x) ; c.g() ;
    D d = new D() ; System.out.println("Objet d:"); d.f(car1) ; c.f(y) ;  d.g();
    E e = new E() ; System.out.println("Objet e:");
    e.f(car1) ; e.f(y) ; e.g(); e.g(car2);
    F ff = new F() ; System.out.println("Objet ff:");
    ff.f(car1) ; ff.f(y) ; ff.g(); ff.g(car2);
  }
}
```

Exercice 2 : (7 points)

Réalisez le code de la fonction suivante :

`static void rotate(int[] a, int d) {...}`

qui effectue une rotation sur les éléments du tableau **a[]** spécifié sur une distance **d**.

Par exemple, si a[] est {60, 61, 62, 63, 64, 65, 66, 67, 68, 69}, alors l'appel rotate(a, 3) le remplace par {67, 68, 69, 60, 61, 62, 63, 64, 65, 66} et rotate(a, -1) le remplace par {61, 62, 63, 64, 65, 66, 67, 68, 69, 60}.

Exercice 3 : (7 points)

Réalisez un programme java composé de :

- une superclasse **ExpressionAlgebrique** contiendrait un attribut **exp** de type **String** qui peut prendre un des trois formes suivantes "2.2 + 5" ou "7 * 2.54" ou "2487 / 5" (à titre d'exemple) et une méthode abstraite **evalue()** qui retourne un **double**,
- trois sous-classes concrètes **Addition**, **Multiplication**, **Division**, qui doivent redéfinir la méthode **evalue()** pour évaluer l'expression **exp** qui représente respectivement une addition, une multiplication ou une division.

بالتوفيـــق. . .

Exercice 1 : (6 points)

Quels résultats fournit le programme suivant ?

```
Objet a:
A==>f(double 4.0 )
A==>g()
Objet b:
A==>f(double 4.0 )
A==>g()
Objet c:
C==>f(char='A')
C==>f(double=4.0)
A==>g()
Objet d:
C==>f(char='A')
C==>f(double=6.0)
D==>g()
Objet e:
E==>f(char='A')
E==>f(double=6.0)
A==>g()
E==>g(char='Z')
Objet ff:
E==>f(char='A')
F==>f(double=6.0)
F==>g()
E==>g(char='Z')
```

Exercice 2 : (7 points)

```
public static  void rotate(int[] t,int d){
        if(d>0){
            for (int i=1;i<=d;i++){
                int temp=t[t.length-1];
                for(int j=t.length-1;j>0;j--){
                    t[j]=t[j-1];
                }
                t[0]=temp;
            }
        }
        else{
            for (int i=1;i<=-d;i++){
                int temp=t[0];

                for(int j=0;j<t.length-1;j++){
                    t[j]=t[j+1];
                }
                t[t.length-1]=temp;
            }
        }
    }
```

Exercice 3 : (7 points)

```
class Division extends ExpressionAlgebrique{

  public Division(String e) {
    super(e);
  }

  @Override
  double evolue(){
    int  posOp=exp.indexOf(":");
    return Double.parseDouble(exp.substring(0, posOp))/Double.parseDouble(exp.substring(posOp+1));
  }
```

```
}

class Addition extends ExpressionAlgebrique{

   public Addition(String e) {
      super(e);
   }
   @Override
   double evolue(){
      int  posOp=exp.indexOf("+");
      return Double.parseDouble(exp.substring(0, posOp))+Double.parseDouble(exp.substring(posOp+1));
   }
}

class Multiplication extends ExpressionAlgebrique{

   public Multiplication(String e) {
      super(e);
   }
   @Override
   double evolue(){
      int  posOp=exp.indexOf("*");
      return Double.parseDouble(exp.substring(0, posOp))*Double.parseDouble(exp.substring(posOp+1));
   }
}

public abstract class ExpressionAlgebrique {
   String exp;
   public ExpressionAlgebrique(String e){
      exp=e;
   }
   abstract double evolue();

   public static void main(String t[]){
      ExpressionAlgebrique[] ea =new ExpressionAlgebrique[3];
      ea[0]=new Division("5.257:2.52");
      ea[1]=new Addition("5.257+2.52");
      ea[2]=new Multiplication("5.257*2.52");
      System.out.println("5.257:2.52="+ea[0].evolue());
      System.out.println("5.257+2.52="+ea[1].evolue());
      System.out.println("5.257*2.52="+ea[2].evolue());
   }
}
```

le main est facultatif,
juste à titre indifcatif