Exercice sur la Pile

```
interface Pile {
    boolean estVide();
    void empiler(Object x);
    Object depiler();
}
```

```
class PileTab implements Pile {
    Objet[] tab = new Object[100];
    int n = 0;
    public boolean estVide() {return n == 0;}
    public void empiler(Object val) {tab[n++] = val;}
    public Object depiler() {return tab[--n];}
}
```

```
class PPile{
  public static void main(String[] args) {

    Pile unePile = new PileTab();
    unePile.empiler("A");
    unePile.empiler("B");
    unePile.empiler(5);
    System.out.println(((PileTab) unePile).n);
    for (int i=0;i<((PileTab) unePile).n;i++)
      System.out.println(((PileTab) unePile).tab[i]);
    Object o;
    while (!unePile.estVide()){
      o=unePile.depiler();
      System.out.println(o);      }
  }
}
```

Exemple sur les getters et les setters

```
class Employe {
        private int nSS;
        private String nom;
        private int age;

        public int getNSS(){return nSS;}
        public void setNSS(int n){ nSS=n;}

        // à compléter pour tester les autres attributs nom et age

}
```

```
public class TestEmploye{
  public static void main(String args[]){
    Employe e= new Employe();
    System.out.print(e.getNSS());
    e.setNSS(5);
    System.out.print(e.getNSS());

  }
}
```

Exercice de la classe Singleton

```
public class A
{
        private static A singleObject;
        private String a;
        public String getA(){return a;         }

        private A(String a){ this.a = a;}

        public static A getInstance(String a){
                if (singleObject == null){
                        singleObject = new A(a);
                }
                return singleObject;
        }

   }
```

```
   public class TestA
   {
        public static void main(String args[]){
        A a;
        a = A.getInstance("Signleton");
        System.out.println("This is the " + a.getA()  );

        A b = A.getInstance("non singleton");
        System.out.println("This is the " +b.getA() );
        }
   }
```