```
interface reproductible{    reproductible reproduire(); }

public abstract class Plante implements reproductible{
    private double  taille;
     void grandir(int n){        taille+=n;      };

    Plante(double taille){      this.taille=taille;     }
    public abstract reproductible reproduire();     }

class Rosier extends Plante{
String couleur;
    void grandir(int n){        super.grandir(n*2);     }
    Rosier(double taille, String couleur) {
        super(taille);
        this.couleur=couleur;   }

    public reproductible reproduire() {     return new Rosier(0, couleur);  } }//end class
    Rosier

class Olivier extends Plante{   List<Fruit> mesFruits;
    Olivier(double taille) {
        super(taille);
        mesFruits=new ArrayList();  }

    Olivier(){      this(0); }

    void grandir(int n) {       super.grandir(n/2);     }

    public reproductible reproduire() {     return new Olivier();   }
    void ajouterFruit(Fruit f){         mesFruits.add(f);   } } // end class Olivier


class Jardin{
    List<Plante> pls;
    void ajouterPlante(Plante p){ pls.add(p);}
    void reproGandirTous(){
        for(int i=0;i<pls.size();i++){
            pls.get(i).grandir(10); // si jardin de Reproductibles alors
            ((Plante)pls.get(i)).grandir(10);
            pls.get(i).reproduire();
            if (pls.get(i) instanceof Olivier) ((Olivier)pls.get(i)).ajouterFruit(new cerise
            ());        }   } }

class cerise implements Fruit{  public void grossir(int poids) {    }
    public void murir(int degre) {  } }
```

Corrigé examen Licence Informatique Univ -Guelma - Dept Informatique D. Bourouaieh

Exercice

```
public class Plaque {
    private int longueur, largeur, poids;
    public Plaque(int longueur, int largeur, int poids) {
        this.longueur = longueur;
        this.largeur = largeur;
        this.poids = poids;     }

    Plaque(int a){          this(a,a,a);    }
    int getLongueur(){ return longueur;}
    int getLargeur(){return largeur;}
    void etirerLongueur(int dist){longueur+=dist;}
    void etirerLageur(int dist){largeur+=dist;}
    void etirer(int dist){      etirer(dist, dist);         }
    void etirer(int d1, int d2){        etirerLongueur(d1);         etirerLageur(d2);   }
    Plaque clonePlaque(){ return new Plaque(longueur, largeur, poids);}

    boolean memeLargeur(Plaque p){ return largeur==p.getLargeur();}

    Plaque collePlaque(Plaque p){
        if(memeLargeur(p)) return  new Plaque(longueur+p.getLongueur(), largeur+p.getLargeur
        (), poids+p.getPoids());
        else return clonePlaque();  }
    boolean compare(Plaque p){return (longueur==p.getLongueur())&&(largeur==p.getLargeur());}
    private int getPoids() { return poids;  } }
```

```
public class Structure {
List<Plaque> mesPlaques;

Structure(List<Plaque> lp){     mesPlaques=lp; }
void coller(int i, int j){  mesPlaques.add(mesPlaques.get(i).collePlaque(mesPlaques.get(j)));}

boolean compare(int i, int j){ return mesPlaques.get(i).compare(mesPlaques.get(j)); }

Plaque collerTout(){
    Plaque p=mesPlaques.get( );//il faut verifier size()
    for(int i= ; i< mesPlaques.size(); i++) p.collePlaque(mesPlaques.get(i));
    return p; }     } //end class Structure

// classe de teste

public class Test {
    public static void main(String[] args) {
List<Plaque> l=new ArrayList<Plaque>();
        l.add(new Plaque( , , )); l.add(new Plaque( , , )); l.add(new Plaque( , , ,4
        ));
        Structure s=new Structure(l);
        s.coller( , );      s.compare( , );       Plaque p=s.collerTout();    }}
```

Exercice

```
interface Fruit{
    void grossir(int poids);
    void murir(int degre); }
```