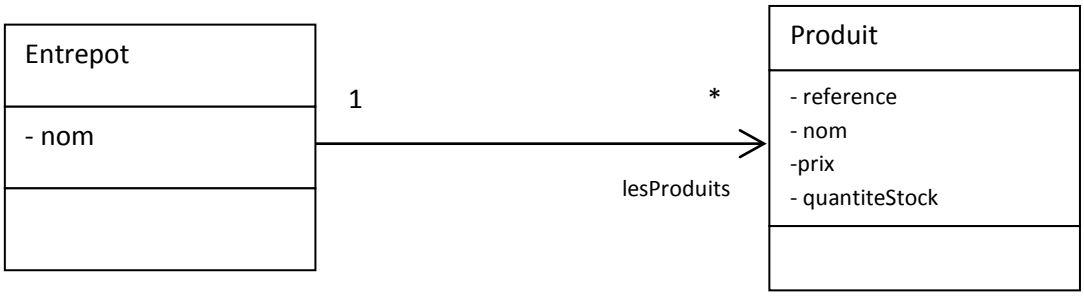
BLOC 2 – Ch 10 - Exercice Burotic93 (collections, associations)

Burotic93 est un distributeur de fournitures de bureau. Un programme écrit en Java permet de gérer les produits et les stocks. Un produit est caractérisé par sa référence, son nom, son prix et par la quantité de ce produit restante en stock.

```
public class Produit
{
    private String reference;
    private String nom;
    private float prix;
    private int quantiteStock;

    // constructeur avec 4 paramètres
    public Produit(String r, String n, float p, int q)
    {
          this.reference = r;
          this.nom = n;
          this.prix = p;
          this.quantiteStock = q;
    }

    // Accesseurs pour reference
    public String getReference()
    {     return reference;     }

    public void setReference(String uneRef)
    {     reference = uneRef;     }

    // Idem pour les autres attributs :
    public String getNom() { … }
    public void setNom(String unNom) { … }
    public float getPrix() { … }
    public void setPrix(float unPrix) { … }
    public int getQuantiteStock() { … }
    public void setQuantiteStock(int uneQuantite)  { … }

    // retourne une description du produit sous forme de String
    public String toString()
    {
        return "produit : ref " + this.reference + ", " + this.nom + ", " +
this.prix + " eur., qte stock : " + this.quantiteStock;
    }
}
```

Les produits sont stockés dans un entrepôt. On vous donne le diagramme de classes partiel suivant :



Le code de la classe **Entrepot** est le suivant :

```
public class Entrepot
{
    // le nom de l'entrepôt
    private String nom;

    // liste des produits stockés
    private ArrayList<Produit> lesProduits = null;

    // Constructeur
    public Entrepot(String unNom)
    {
        this.nom = unNom;
        // instancie la liste des produits :
        lesProduits = new ArrayList<Produit>();
    }

    // ajoute à la liste le produit passé en paramètre
    public void ajouteProduit(Produit unProduit)
    {
        lesProduits.add(unProduit);
    }

    // retourne le nombre de produits de la liste
    public int getNombreProduits()
    {
        return lesProduits.size();
    }
}
```

On vous fournit une documentation ci-dessous.

Documentation 1 : Exemple d'utilisation d'une collection (**ArrayList**)

L'exemple ci-dessous permet de manipuler une collection de chaînes de caractères. Le principe est le même quel que soit le type des éléments.

```
ArrayList<String> mesChaines;          // déclaration d'une collection de chaînes de caractères
mesChaines = new ArrayList<String>();  // instanciation de la collection
mesChaines.add("un");                  // ajout d'une chaîne à la collection
mesChaines.add("deux");
mesChaines.add("trois");
for (String uneChaine : mesChaines) {  // parcours de la collection
      System.out.println(uneChaine);   // affichage de l'élément courant
}
mesChaines.remove(1);                  // suppression du 2ème élément (indice 1)
mesChaines.remove("trois");            // suppression de l'objet « trois »
System.out.println(mesChaines.get(0)); // affichage du 1er élément (indice 0)
System.out.println(mesChaines.size()); // affichage du nombre d'éléments de la collection
```

Travail à faire

Question 1 : Ecrire le code d'une méthode **main** d'une classe **Program** qui fait les choses suivantes :

- instancie un entrepôt nommé « Entrepot Marseille » et l'affecte à la variable **unEntrepot** ;
- instancie trois produits p1, p2 et p3 (inventez des valeurs) ;
- ajoute ces trois produits à l'entrepôt ;
- obtient et affiche le nombre de produits de l'entrepôt.

Question 2 : Ecrire le code de la méthode **toString** de la classe **Entrepot** qui retourne, sous forme de **String**, une desciption complète de l'entrepôt, c'est-à-dire son nom et la description complète de ses produits. La signature de la méthode est :

```
public String toString()
```

Question 3 : Ecrire le code de la méthode **getQuantiteStockTotale** de la classe **Entrepot** qui retourne la quantité totale de produits restants en stock. La signature de la méthode est :

```
public int getQuantiteStockTotale()
```

Question 4 : Ecrire le code de la méthode **rechercherProduit** de la classe **Entrepot**, qui permet de retrouver un produit dans la liste des produits de l'entrepôt, à partir de sa référence. Cette méthode a la signature suivante :

```
public Produit rechercheProduit(String ref)
```

Elle a un paramètre :

- la référence du produit à rechercher.

elle retourne :

- le produit correspondant à la référence passée en paramètre, si ce produit existe,
- **null** sinon.

Question 5. On souhaite connaître les produits de l'entrepôt qui sont en rupture de stock, c'est-à-dire ceux pour lesquels la quantité en stock est égale à 0.

Travail à faire. Ecrire dans la classe **Entrepot** la méthode **getProduitsSansStock()** qui retourne la liste de tous les produits en rupture de stock de l'entrepôt (compléter le code ci-dessous).

```
// Code de la méthode à compléter
public ArrayList<Produit> getProduitsSansStock()
{
    // instantication de la liste des produits sans stock
    ArrayList<Produit> listeProd = new ArrayList<Produit>();

    // parcours de la liste de tous les produits

    for (                                                    )
    {
        // si un produit a une quantité en stock à 0, on l'ajoute à la liste
        …
    }

    // on retourne la liste
    return listeProd;
}
```

Question 6. On souhaite obtenir la liste des produits de l'entrepôt qui ont une quantité en stock supérieure à une certaine quantité donnée.

Travail à faire. Ecrire dans la classe **Entrepot** la méthode **getProduitsEnStock(int quantiteMini)** qui retourne la liste des produits de l'entrepôt qui ont une quantité en stock supérieure au paramètre **quantiteMini**. La signature de la méthode est :

```
public ArrayList<Produit> getProduitsEnStock(int quantiteMini)
```