**Université Mohammed Premier**

**Faculté des Sciences**

**Département d'Informatique**
**Oujda**

---

# Exercices et Examens corrigés

# en POO-JAVA

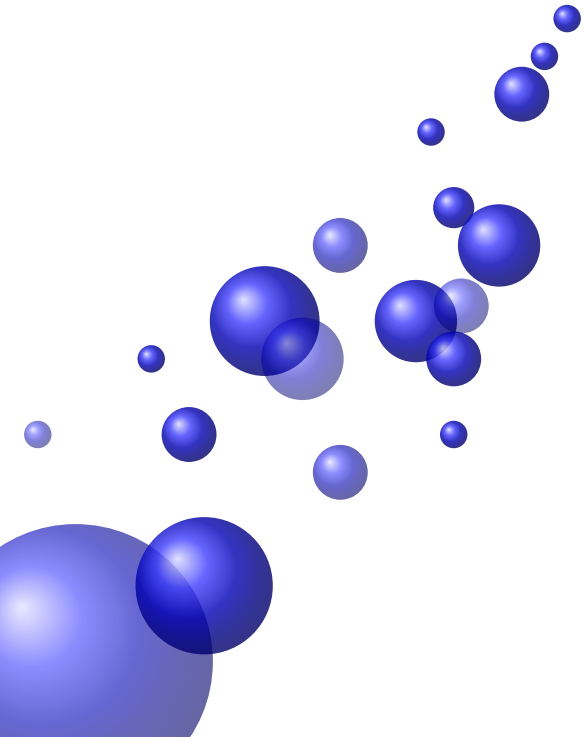
---

**Filière : SMI**
**Semestre : 5**

Pr. Mohammed GABLI

https://sites.google.com/ump.ac.ma/gabli/

Année universitaire 2021/2022



# Préambule

Ce polycopié contient un ensemble d'exercices et applications en Java ainsi que leurs corrections.
La plupart des applications sont tirées des examens de la programmation orientée objet en Java de 2018 à 2022 pour les étudiants de la Filière Sciences Mathématiques et Informatique (SMI, Semestre 5) de l'Université Mohamed Premier, Oujda.
Avant de lire ce poplycopié, le lecteur est invité à consulter mon cours intitulé `Programmation Orientée Objet en JAVA` et publié dans mon site :
`https://sites.google.com/ump.ac.ma/gabli/teaching`.
Il est aussi fortement recommandé de réaliser et répondre aux exercices avant de lire et analyser la correction.

# Table des matières

# Chapitre 1

# Exercices (du cours)

### Exercice 1

Qu'affiche le code suivant ? (donnez d'abord l'affichage, puis expliquez brièvement pourquoi).

```
class A{
    static int x = 2;
    int y = 3;
    public void message(){
      System.out.println("Methode message de la classe A");
    }
    public static void g(){
      System.out.println("Methode g() de la classe A");
    }
    public int calcul(){
        return (x + y);
    }
}
class B extends A{
    public void message(){
      System.out.println("Methode message de la classe B");
    }
    public void f(){
```

```
      System.out.println("Methode f() de la classe B");
    }
    public static void g(){
      System.out.println("Methode g() de la classe B");
    }
}
public class Exam2018{
  public static void main(String[] args) {
    A a = new A();
    B b = new B();
    a.message();
    b.message();
    b.f();
    a = new B();
    a.message();
    a.g();
    a.x=5; a.y=7;
    System.out.println("Apres calcul: " + b.calcul());
 }
}
```

**Exercice 2**

Qu'affiche le code suivant ?

```
class A{
    static int x = 5;
    int y = 4;
    public static void test(int [][]tableau, int x) {
      for (int i = 0; i < tableau.length; i++)
          for (int j = 0; j < tableau[i].length; j++)
                tableau[i][j] = 5;
```

```
            x = 10;
        }
    public int calcul(){
        return (x + y);
    }
  }
public class Exam2019{
  public static void main(String[] args) {
    int[][] tab = {{1,2,3},{4,5}};
    A a = new A();
    A b = new A();
    System.out.println(tab.length);
    a.test(tab,tab[0][0]);
    for (int i = 0; i < tab.length; i++)
        for (int j = 0; j < tab[i].length; j++)
            System.out.print(tab[i][j]+"-");
    a.x=12; a.y=3;
    System.out.println("\nApres calcul: " + b.calcul());
   }
 }
```

**Exercice 3**

Soit le programme en Java suivant :

```
class Vehicule{
  public void afficher(){
    System.out.println("J'ai quatre roues");
  }}
class Voiture extends Vehicule{
  public void afficher(){
    System.out.println("Je suis une voiture");
```

```
    }}
class Transport{
  public void demarrer(Vehicule v){
    System.out.println("Un vehicule demarre");
    v.afficher();
    }
  public void demarrer(Voiture v){
    System.out.println("Une voiture demarre");
    v.afficher();
    }}
public class EssaiVehicule{
  public static void main(String args[]){
    Transport.demarrer(new Vehicule());
    Transport.demarrer(new Voiture());
    Vehicule A = new Voiture();
    Transport.demarrer(A);
    }}
```

La compilation de ce code génère l'erreur suivante : *error : non-static method demarrer(Vehicule) cannot be referenced from a static context Transport.demarrer(new Vehicule()).*

1. Corrigez l'erreur (les erreurs) ;
2. Qu'affiche le programme ci-dessus après correction ?

**Exercice 4**

Qu'affiche le code suivant ?

```
class Etudiant {
    private String nom, prenom;
    public Etudiant(String x,String y){
```

```
        this.nom = x;
        this.prenom = y;
    }
    public void afficher() {
       System.out.println("Nom: " +
       nom + "et prenom: " + prenom );
    }
  }
public class Examen {
    public static void main(String[] argv) {
    Etudiant []tab1 = {new Etudiant("a","b"),
                       new Etudiant("aa","bb")};
    Etudiant []tab2 = {new Etudiant("c","d"),
                       new Etudiant("cc","dd")};
    Etudiant []tab3 = {new Etudiant("c","d"),
                       new Etudiant("cc","dd")};
    tab1 = tab2;
    System.out.println(tab1.length);
    System.out.println(tab1.equals(tab2));
    System.out.println(tab3.equals(tab2));
    tab2[0] = new Etudiant("aaa","bbb");
    tab1[0].afficher();
    tab2[0].afficher();
    tab3[0].afficher();
    }
}
```

**Exercice 5**

Qu'affiche le programme en Java suivant ?

```
class A{
```

```
    static int x = 0;
    int y = 0;
    final int z;
    public A(int z){
      x++;
      y++;
      this.z = z;
      }
    public void afficher() {
      System.out.println("x:"+x+ " y:" +y+ " et z:" + z);
      }
    public void test(A a, int x){
      x = 8;
      a.y = 9;
   }
  }
public class Exam2019{
  public static void main(String[] args) {
    A a = new A(2);
    A b = new A(3);
    a.afficher();
    b.afficher();
    a.test(a,a.x);
    b.test(a,a.x);
    a.afficher();
    b.afficher();
   }
 }
```

**Exercice 6**

Qu'affiche le programme en Java suivant ?

```
class Publication{
   int a = 5;
   public void afficher() {
      System.out.println("chercher une publication");}
   public void m(){
      System.out.println("a ="+ a);}
  }
class Livre extends Publication{
   int a = 6;
   public void afficher() {
      System.out.println("chercher un livre.");
      super.afficher();
      }
   public void m(){
      System.out.println("a ="+ a);}
  }
class Manuel extends Livre{
   int a = 7;
   public void afficher(){
      System.out.println("chercher un manuel.");
      super.afficher();
      }
   public void m(){
      System.out.println("a ="+ a);}
  }
public class Test{
  public static void main(String [] args){
      Publication pubBook = new Livre();
      Livre book = new Manuel();
```

```
        pubBook.afficher();
        book.afficher();
        pubBook.m();
        book.m();
      System.out.println("La somme est: "+pubBook.a+book.a);
        }
  }
```

**Exercice 7**

Soit le programme en Java suivant :

```
 class Etudiant {
    private String nom;
    private static int num = 1;
    public Etudiant(String nom){
      this.nom = nom;
      this.num++;
    }
    public void afficher() {
      System.out.println("Nom: "+nom+ " et numero: "+num);
    }
  }
public class Exam2021 {
  public static void main(String[] argv) {
    Etudiant []tab = new Etudiant[3];
    tab[0].afficher();
    Etudiant []tab1 ={new Etudiant("a"),new Etudiant("b")};
    Etudiant []tab2 ={new Etudiant("c"),new Etudiant("d")};
    Etudiant []tab3 ={new Etudiant("c"),new Etudiant("d")};
    tab1 = tab2;
    System.out.println(tab1.length);
```

```
    System.out.println(tab1.equals(tab2));
    System.out.println(tab3.equals(tab2));
    tab2[0] = new Etudiant("aa");
    tab1[0].afficher();
    tab3[0].afficher();
    }
}
```

L'exécution de ce code génère l'erreur suivante : *Exception in thread "main" java.lang.NullPointerException at Exam2021.main(Exam2021.java :15)* .
Corrigez l'erreur puis donnez l'affichage du programme ci-dessus.

**Exercice 8**

Soit le code suivant en Java :

```
public class Exercice {
    static private String msg = null;
    static private int n;
    public Exercice(){
        n = 1;
        if (msg == null) msg = "Rouge";
        afficher();
        }
    public Exercice(Exercice autre){
        this.msg = autre.msg;
        this.n = autre.n;
        }
    private void afficher(){
        System.out.println(n + ". " + msg);
        if (!msg.equals("Vert")){
                msg = "Vert";
                new Exercice();
```

```
            }
        }
    public static void main(String[] args){
        Exercice x = new Exercice();
        n++;
        x.afficher();
        Exercice y = new Exercice(x);
        n++;
        y.afficher();
        if(x.equals(y)) System.out.println(" egalite ");
        else System.out.println(" pas d'egalite ");
        }
    }
```

1. Qu'affiche le programme précédent ?
2. Ce programme peut-il être compilé si on supprime le mot-clé `static` de la deuxième ligne ? Si oui, qu'affiche-t-il ? Si non, pourquoi ?
3. Mêmes questions en enlevant, cette fois, le mot-clé `static` de la troisième ligne.

# Chapitre 2

# Exercices d'application

### Exercice 1

On désire automatiser la gestion d'une bibliothèque qui comprend des livres et des adhèrents.

**1.** Créez une classe **Personne** qui contient :
- les attributs **privés** : **nom**, **prenom** et **age** ;
- un constructeur pour initialiser les différents attributs ;
- une méthode **afficher()** qui affiche le nom, le prénom et l'age.

**2.** Créez une classe **Adherent** qui hérite de **Personne** et qui :
- ajoute l'attribut **numAdherent** ;
- blackéfinit la mèthode **afficher()**.

**3.** Créez une classe **Auteur** qui hérite de **Personne** et qui :
- ajoute l'attribut **numAuteur** ;
- blackéfinit la mèthode **afficher()**.
- le programme génère une exception si l'utilisateur veut instancier un objet Auteur avec une valeur négative de **numAuteur**.

**4.** Un livre est définit par un numèro ISBN (entier), un titre et est écrit par au plus 4 auteurs.
- créez la classe **Livre** ;
- ajoutez une mèthode **afficher()** qui affiche le ISBN, le titre et les informations des auteurs.

**5.** Créez une classe **Bibliothèque** qui contient une méthode **main()**, dans laquelle :
- déclarez (instanciez) un adhèrent ;
- instanciez un livre qui est écrit par deux auteurs ;
- affichez les informations de l'adhèrent et du livre.

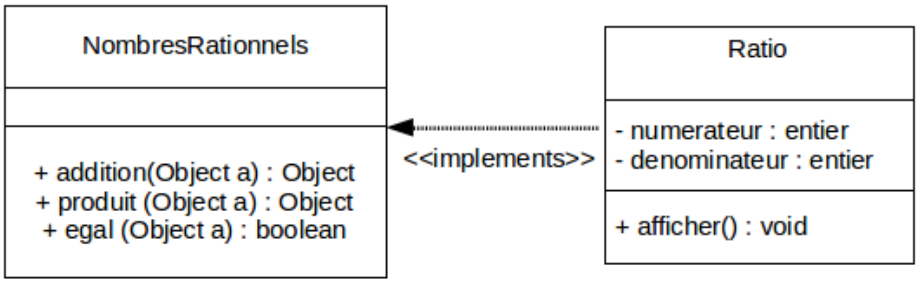
### Exercice 2

Un nombre rationnel est un nombre qui s'écrit sous la forme $a/b$, où a et b sont des entiers.

1. Écrivez une interface **NombresRationnels** qui contient les mèthodes suivantes (voir figure) :
   - `addition()` qui a un seul argument de type `Object` et qui retourne un `Object`. Elle retourne la somme du nombre rationnel courant et celui passé en paramètre ;
   - `produit()` qui a un seul argument de type `Object` et qui retourne un `Object`. Elle retourne le produit du nombre rationnel courant et celui passé en paramètre ;
   - `egale()` qui a un seul argument de type `Object` et qui renvoie true si le nombre rationnel courant et celui passé en paramètre sont égaux (et renvoie false sinon).
2. Écrivez la classe **Ratio** qui implémente l'interface **NombresRationnels** et qui contient :
   - deux attributs `numerateur` et `denominateur` de type entier ;

— un constructeur qui a deux arguments. **Attention :** le programme lance une exception de type **ArithmeticException** si le denominateur est égal à 0 ;
— une implèmentation des trois mèthodes : `addition()`, `produit()` et `egale()` ;
— une mèthode **afficher()** qui retourne une chaine de caractère de la forme $a/b$.

3. Écrivez une classe **Test** qui contient la mèthode **main()**. Dans laquelle :
— déclarez et affichez les deux rationnels Ratio(2,3) et Ratio(5,7) ;
— Appliquez les mèthodes `addition()`, `produit()` et `egale()` à ces deux nombres, et affichez les résultats de chaque opération.
— déclarez le nombre rationnel Ratio(1,0) ;




## Exercice 3

**1.** Créez une classe **Vehicule** qui contient :
— les attributs : `immatriculation`, `marque` et `couleur` (tous de type chaine de caractères) ;
— un constructeur pour initialiser les différents attributs ;
— une méthode `afficher()` qui affiche les trois attributs.

**2.** Créez une classe **Roue** qui est caractérisée par le type (chaine de caractères), la largeur (réel) et la hauteur (réel).
— ajoutez un constructeur de trois arguments qui déclenche une **exception** si la largeur ou la hauteur est négative.
— ajoutez une mèthode `afficher()` qui affiche les informations d'une roue.

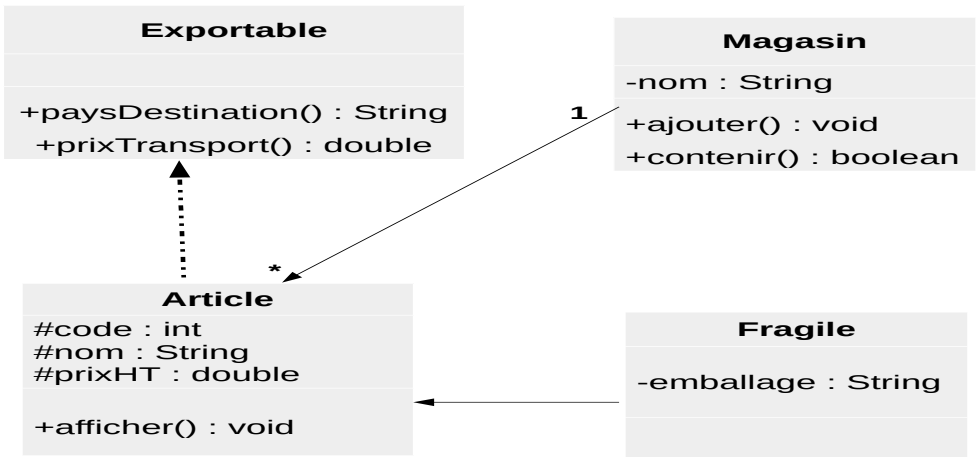
**3.** Créez une classe **Voiture** qui hérite de **Vehicule** et qui :

— ajoute l'attribut `Roue` ;
— blackéfinit la mèthode `afficher()`.

**4.** Créez une classe **Test** qui contient une méthode **main()**, dans laquelle :
— déclarez une roue ;
— instanciez une voiture qui contient la roue ainsi déclarée ;
— affichez les informations de la voiture.

### Exercice 4

La phase d'analyse et de conception du problème de la gestion des ventes d'articles dans un magasin a conduit au diagramme de classes ci-dessous.




1. Programmez l'interface **Exportable** telle qu'elle est indiquée dans le diagramme de classes.
2. Écrivez la classe **Article** qui implémente l'interface **Exportable** et qui contient :
   — trois attributs : un code qui s'incrèmente automatiquement, un nom et un prix hors taxe (prixHT) ;

— un constructeur qui a deux arguments (nom et prixHT) et qui initialise les trois attributs ;
— le programme doit lancer une exception si le prix hors taxe est négatif ou nul ;
— pour implémenter la mèthode *paysDestination()*, on demande à l'utilisateur d'entrer le pays voulu ;
— la mèthode *prixTransport()* retourne 5% du Prix Hors Taxes de l'article concerné ;
— une mèthode *afficher()* qui affiche le code, le nom, le pays et le prix total. Ce dernier est la somme du prixHT et le prix de transport.

3. Écrivez la classe **Fragile** qui hérite de la classe **Article** et qui contient :
   — un attribut emballage (de plus) ;
   — une blackéfinition de la méthode *prixTransport()* pour que le prix de transport d'un article fragile devienne deux fois le prix de transport d'un Article normal.

4. Écrivez la classe **Magasin** qui contient un nom et un ensemble d'articles (le nombre n'est pas défini).
   — Ajoutez une méthode *ajouter()* qui permet d'ajouter un article au magasin ;
   — Ajoutez une méthode *contenir()* qui vérifie si un article est disponible au magasin.

5. Écrivez la classe principale **Test** qui contient la mèthode *main()*.
   — Déclarez un tableau de quatre articles. Puis, instanciez les deux premiers éléments par des articles et les deux derniers par des fragiles ;
   — Déclarez un magasin qui contient les trois premiers éléments du tableau ;
   — Vérifiez si le magasin contient le quatrième élément du tableau, puis ajoutez ce dernier élément au magasin.

**Exercice 5**

On désire gérer la sélection des candidats dans un concours. Pour cela, on définit les informations nécessaires des candidats autorisés à participer à ce concours :

- **Nom** : Nom du candidat
- **Prénom** : Prénom du candidat
- **Numéro** : Numéro du candidat
- **Age** : L'age du candidat. Il ne doit pas dépasser 30 ans.

Les fonctionnaires, eux aussi, peuvent participer à ce concours s'ils remplissent les conditions de participation, en particulier, la condition d'age.

**1.** Créez la classe **Candidat** qui contient les 4 attributs citès ci-dessus.

- Ajoutez un constructeur qui a quatre arguments ;
- Ajoutez une méthode **afficher()** qui affiche toutes les informations d'un candidat.

**2.** Créez une classe **Fonctionnaire** qui hérite de la classe **Candidat** et qui :

- ajoute aux propriétés d'un candidat le numéro de somme (Num_Somme en entier).
- blackéfinit la mèthode **afficher()**.

**3.** Modifiez la classe **Candidat** en ajoutant une exception qui sera déclenchée si l'age associé au candidat est supérieur à 30.

**4.** Créez une classe **Concours** qui contient une méthode **main()**, dans laquelle :

- déclarez (instanciez) un candidat ;
- instanciez un fonctionnaire ;
- déclarez une collection de type ArrayList ;
- ajoutez le candidat et le fonctionnaire à cette collection ;

— afficher les informations du candidat et du fonctionnaire en utilisant cette collection.

**Exercice 6**

**Note :** `tous les attributs doivent être déclarés privés. Ajoutez les getters nécessaires pour le bon fonctionnement du programme.`

1. Écrivez une classe **Produit** qui a un libellé (un nom) et un prix ;
   — ajoutez un constructeur qui initialise les deux arguments ;
   — ajoutez une mèthode `afficher()` qui affiche les informations d'un produit.

2. Écrivez une classe **Commande** qui a un numéro, une quantité et un produit et qui contient :
   — un constructeur de trois arguments qui déclenche une exception si la quantité est négative ou nulle ;
   — une mèthode `afficher()` qui affiche les informations d'une commande.

3. Écrivez une classe **Personne** définie par son nom et son age et qui contient un constructeur de deux arguments et une mèthode `afficher()`.

4. Écrivez une classe **Client** qui hérite de la classe **Personne** et qui contient au plus 3 commandes ;
   — ajoutez un constructeur convenable ;
   — ajoutez une mèthode `afficher()` qui affiche les informations d'un client ;
   — ajoutez une mèthode `payer(Commande[] commande)` qui retourne le prix total payé par le client ($prixUneCommande = prix * quantite$)

5. Écrivez une classe **Test** qui contient la mèthode **main()**. Dans laquelle :
   — déclarez un tableau de commandes de taille 3 ;
   — initialisez les attributs de chaque commande en les saisissant au clavier ;
   — déclarez un client qui demande ces trois commandes ;
   — affichez le client ;
   — affichez le prix total payé par le client.

**Exercice 7**

Nous voulons écrire un programme en Java pour la gestion des réservations des chambres d'un hotel.

1. Écrivez une classe **Client** qui a quatre atributs : id, nom, prenom et pays ;
   — ajoutez un constructeur qui initialise les quatre attributs ;
   — ajoutez une mèthode `afficher()` qui affiche les informations d'un client.

2. Écrivez une classe abstraite **Chambre** qui a quatre attributs : id_chambre qui s'incrèmente automatiquement, num_etage, nombre_lits et disponible (attribut booléen initialisé par true pour dire que la chambre est disponible). Cette classe contient aussi :
   — un constructeur qui a deux arguments (num_etage et nombre_lits) et qui initialise les trois attributs id_chambre, num_etage et nombre_lits ;
   — une mèthode `setDisponible()` qui inverse la valeur de l'attribut *disponible* ;
   — une mèthode abstraite `afficher()`.

3. Écrivez une classe **ChambreDouble** qui hérite de la classe **Chambre** et qui fixe le nombre de lits en 2. Cette classe contient aussi :
   — un constructeur qui a un seul argument : num_etage et qui initialise num_etage et nombre_lits ;
   — une implémentation de la mèthode `afficher()`.

4. Écrivez une classe **Reservation** qui contient un client, une chambre, num_res (numèro de la réservation) et nombre_nuits ;
   — ajoutez un constructeur qui déclenche une exception si la chambre est indisponible ; sinon, le constructeur doit initialiser les quatre attributs et déclarer que la chambre devienne indisponible ;
   — ajoutez une mèthode `afficher()` qui affiche toutes les informations d'une réservation.

5. Écrivez une classe **Hotel** qui contient la mèthode **main()**. Dans laquelle :
   — créez une chambre et deux clients et afficher ces informations ;
   — créez une réservation affectant le premier client à la chambre ;
   — créez une deuxième réservation affectant le deuxième client à la chambre. Qu'est ce que vous constatez ? (donnez le résultat de cette instruction) ;
   — Commentez l'instruction suivante (valide ou non et pourquoi ?) :

```
Chambre a = new Chambre(4,1);
```

**Exercice 8**

L'objectif de cet exercice est de réaliser un code en Java pour gérer les groupes des étudiants d'un établissement donné.

1. Programmez une classe abstraite **Personne** qui contient un seul attribut privé *nom*, un constructeur et une méthode abstraite *afficher()*.
2. Écrivez la classe **Etudiant** qui hérite de **Personne** et qui contient :
   — six notes ;
   — un constructeur qui a un seul argument (nom) et qui demande à l'utilisateur d'entrer les notes de l'étudiant concerné ;
   — le programme doit lancer une exception si la note n'est pas comprise entre 0 et 20 ;
   — une méthode *moyenne()* qui calcule la moyenne des notes d'un étudiant ;
   — une implémentation de la mèthode *afficher()* pour afficher le nom et la moyenne.

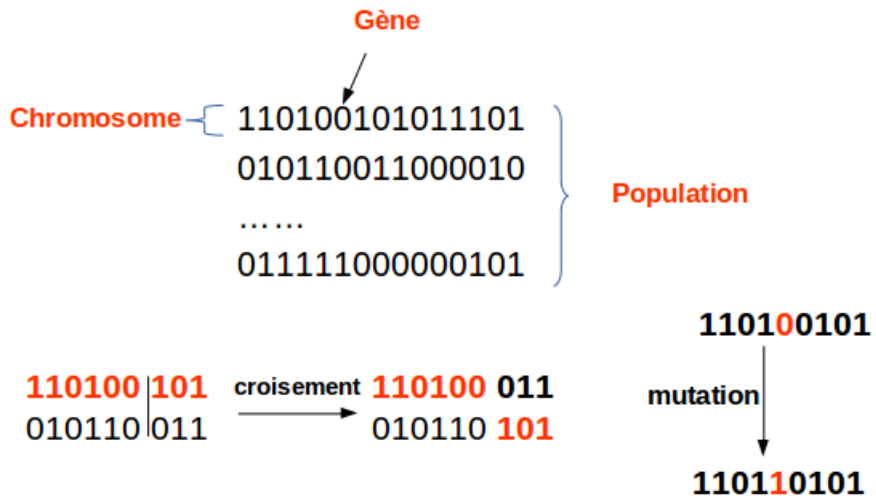
3. Un groupe d'étudiants est un ensemble d'étudiants dont le nombre est inconnu. Écrivez la classe **GroupeEtudiant** qui contient les méthodes suivantes :
   — une méthode *nombre()* qui retourne le nombre d'étudiants dans un groupe ;
   — une méthode *ajouterEtudiant()* qui ajoute un étudiant au groupe.
   — une méthode *chercher(String nom)* qui retourne l'étudiant ayant le nom cherché.
   — une méthode *lister()* qui affiche tous les étudiants d'un groupe.
4. Écrivez la classe principale **Test** qui contient la mèthode *main()*.
   — Créez deux étudiants et un groupe d'étudiants ;
   — ajoutez les deux étudiants au groupe puis affichez la taille de ce groupe ;
   — affichez tous les étudiants de ce groupe ;
   — Vérifiez si le groupe contient l'étudiante *Maryem* en affichant ces informations si elle existe.
5. Soit l'instruction suivante

```
Personne A = new Etudiant("Said");
```

   Est-ce qu'on peut ajouter l'objet **A** au groupe en appliquant la méthode *ajouterEtudiant()* ?
   — expliquez pourquoi si la rèponse est positive.
   — proposez une solution si la rèponse est négative.

### Exercice 9

Les algorithmes génétiques (Genetic Algorithms : GAs) sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisement, mutation, sélection ...
Dans ces algorithmes, une population contient un ensemble de chromosomes, et chaque chromosome contient un ensemble de gènes.
Une mutation consiste simplement à inverser un gène (1 en 0 ou l'inverse).
Pour le croisement, on choisit deux chromosomes et une position à partir de laquelle on échange les gènes de chaque chromosome (voir figure).




1. Écrire une classe **Gene** qui contient un attribut **valeur** initialisé aléatoirement à 0 ou à 1 par un constructeur (sans arguments).
   **Indication :** utiliser $Math.random()$ pour un tirage aléatoire entre 0 et 1.
2. Écrire une classe **Chromosome** qui contient un ensemble de **gènes**. Pour ceci :
   — déclarer un tableau de 100 **gènes** ;
   — un constructeur qui a comme argument le nombre de gènes associé à chaque chromosome et qui initialise (instancie) les différents gènes ;
   — une mèthode **afficher()** pour afficher un chromosome.
3. Écrire une classe **Population** qui contient un ensemble de **chromosomes**. Pour ceci :
   — déclarer un tableau de 100 **chromosomes** ;
   — un constructeur qui a deux arguments : le nombre de chromosomes et le nombre de gènes associé à chaque chromosome, et qui instancie les différents chromosomes ;
   — une mèthode **afficher()** pour afficher une population ;
   — une mèthode **mutation(chromosome v,int a)** pour muter le gène $a$ du chrom $v$ ;
   — une mèthode **croisement(chromosome u, chromosome v, int a)** pour croiser les chromosomes $u$ et $v$ selon la position $a$.
4. Écrire une classe **GA** qui contient la mèthode **main()**.

— entrer une population de 10 chromosomes, chacun de 20 gènes et afficher la population ;
— croiser le premier et dernier chromosomes selon l'indice 15 et muter le gène 6 du troisième chromosome ;
— afficher la nouvelle population.

**Exercice 10**

Nous voulons écrire un programme en Java pour simuler le comportement d'un robot .

**1.** Créez une classe abstraite **Robot** qui contient :
— les quatres attributs `privés` : **nom**, **x, y** (x et y sont deux entiers pour définir la position) et **direction** (qui prend uniquement une des valeurs : "Nord", "Est", "Sud" ou "Ouest") ;
— un constructeur pour initialiser les différents attributs. Le programme génère une exception si l'utilisateur entre une direction différente des quatre valeurs citées ci-dessus ;
— une méthode **avancer()** qui incrémente x en allant vers l'Est et le décrémente dans le sens contraire, et qui incrémente y en allant vers le Nord et le décrémente dans le sens contraire ;
— une méthode **droite()** pour tourner à droite de 90° pour changer de direction (si sa direction était "Nord" elle devient "Est", $\cdots$) ;
— une méthode **afficher()** pour afficher les informations du robot ;
— trois méthodes abstraites **gauche()**, **demiTour()** et **avancer(int pas)**.

**2.** Créez une classe **RobotNG** qui hérite de **Robot** et qui implémente les méthodes **gauche()** pour tourner à gauche, **demiTour()** pour faire un demi-tour et **avancer(int n)** pour avancer de $n$ pas, en respectant les consignes suivantes :
— en appellant seulement les anciennes méthodes pour implémenter le nouveau

comportement ;

— donnez, sans écrire le code, une deuxième solution plus efficace qui change directement l'état de l'objet sans faire appel aux anciennes méthodes.

**3.** Créez une classe **RobotExam** qui contient une méthode **main()**, dans laquelle :

— déclarez un tableau de trois robots ;

— le premier robot avance puis tourne à droite ;

— le deuxième robot avance de trois pas puis tourne à droite ;

— le troisième robot avance de 7 pas, tourne à gauche et fait un demi-tour ;

— affichez les informations des trois robots.

# Chapitre 3

# Correction d'exercices du cours

### Correction d'exercice 1

Le code affiche :
Methode message de la classe A
Methode message de la classe B
Methode f() de la classe B
Methode message de la classe B
Methode g() de la classe A
Apres calcul : 8

### Correction d'exercice 2

Le code affiche :
2
5-5-5-5-5-
Apres calcul : 16

### Correction d'exercice 3

```
class Vehicule{
    public void afficher(){
        System.out.println ("J'ai quatre roues");
```

```
  }
}
class Voiture extends Vehicule{
    public void afficher(){
        System.out.println ("Je suis une voiture");
    }
}
class Transport{
    public static void demarrer(Vehicule v){
        System.out.println("Un vehicule demarre");
        v.afficher();
    }
    public static void demarrer(Voiture v) {
        System.out.println ("Une voiture demarre");
        v.afficher();
    }
}
public class EssaiVehicule{
    public static void main(String args[]){
        Transport.demarrer(new Vehicule());
        Transport.demarrer(new Voiture());
        Vehicule A = new Voiture();
        Transport.demarrer(A);
    }
}
```

Ce code donne comme résultat :

Un vehicule demarre

J'ai quatre roues

Une voiture demarre

Je suis une voiture

Un vehicule demarre
Je suis une voiture

### Correction d'exercice 4

2
true
false
Nom : aaa et prenom : bbb
Nom : aaa et prenom : bbb
Nom : c et prenom : d

### Correction d'exercice 5

x :2 y :1 et z :2
x :2 y :1 et z :3
x :2 y :9 et z :2
x :2 y :1 et z :3

### Correction d'exercice 6

chercher un livre.
chercher une publication
chercher un manuel.
chercher un livre.
chercher une publication
a =6
a =7
La somme est : 56

## Correction d'exercice 7

### Correction

Il faut ajouter l'instruction

```
tab[0] = new Etudiant("Ahmed");
```

avant l'instruction

```
tab[0].afficher();
```

### Affichage

Nom : Ahmed et numero : 2

2

true

false

Nom : aa et numero : 9

Nom : c et numero : 9

## Correction d'exercice 8

1. Le code afiche :
   1. Rouge
   1. Vert
   2. Vert
   3. Vert

   pas d'égalité
2. Oui (pas d'erreur de compilation). Mais à l'exécution, le programme affiche "1. Rouge" un nombre indéfini de fois (jusqu'à ce qu'il y ait dépassement de pile (appels récursifs infinis)).
3. Non, le compilateur signale deux erreurs car les deux instructions $n^{++}$; dans la méthode main n'ont pas de sens. En effet, maintenant $n$ est une variable d'instance (non static) donc elle doit être appelée en utilisant un objet (par exemple : $x.n^{++}$;).

# Chapitre 4

# Correction d'exercices d'application

### Correction d'exercice 1

```
import java.util.Scanner;
class AgeException extends Exception{
    public AgeException(){
        System.out.println("Numero d'auteur ne doit pas
                            etre negatif");
    }
}
class Personne {
    private String nom, prenom;
    private int age;
    public Personne(String x,String y,int z){
        nom = x;
        prenom = y;
        age = z;
        }
    public void afficher() {
        System.out.println("Nom: " +nom + ", prenom: "
                        +prenom + ", age: " +age);
    }
}
```

```
class Adherent extends Personne {
    private int numAdher;
    public Adherent(String x,String y,int z, int l){
        super(x,y,z);
        numAdher = l;
    }
    public void afficher() {
        super.afficher();
        System.out.println( " et numero: " +numAdher);
    }
}
class Auteur extends Personne {
    private int numAut;
    public Auteur(String x,String y,int z, int l)throws
                AgeException{
        super(x,y,z);
        if ( l <= 0 ) throw new AgeException();
        else  numAut = l;
    }
    public void afficher() {
        super.afficher();
        System.out.println( " et numero: " +numAut);
    }
}
class Livre {
    private String titre;
    private int ISBN;
    private Auteur[] auth = new Auteur[4];
    public Livre(String x,int y, Auteur[] au ){
        titre = x;
        ISBN = y;
```

```
        auth = au;
    }
    public void afficher() {
        System.out.println("Livre de titre: " + titre +
            ", ISBN: " +ISBN" + et les auteurs sont: ");
        for(int i =0;i<auth.length;i++){
            auth[i].afficher();
            }
    }
}
public class CorrectionExam2018v2{
  public static void main(String[] args) {
    try{
        Adherent a = new Adherent("a", "b", 20, 1);
        Auteur[] aa = new Auteur[2];
        aa[0] = new Auteur("aa", "ba", 25, 11);
        aa[1] = new Auteur("aaa", "baa", 35, 111);
        Livre b = new Livre("Java",15222,aa);
        a.afficher();
        b.afficher();
        } catch(AgeException e){
            System.out.println(e);
            System.exit(-1);
        }
    }
}
```

**Correction d'exercice 2**

```
interface NombresRationnels{
    Object addition(Object a);
```

```
    Object produit(Object a);
    boolean egale(Object a);
}

class Ratio implements NombresRationnels{
    private int numerateur, denominateur;
    public Ratio(int numerateur, int denominateur){
        if (denominateur == 0) throw  new
        ArithmeticException("Un rationnel ne peut pas avoir
                un denominateur nul");
        else{
            this.numerateur = numerateur;
            this.denominateur = denominateur;
        }
    }
    public Object addition(Object a){
        return new Ratio(
        this.numerateur * ((Ratio)a).denominateur +
        this.denominateur * ((Ratio)a).numerateur,
        this.denominateur * ((Ratio)a).denominateur);
       }
    public Object produit(Object a){
        return new Ratio(
        this.numerateur * ((Ratio)a).numerateur,
        this.denominateur * ((Ratio)a).denominateur);
       }
    public boolean egale(Object a){
      return(this.numerateur * ((Ratio)a).denominateur ==
        this.denominateur * ((Ratio)a).numerateur);
       }
    public void afficher(){
```

```
        System.out.println(numerateur + "/" +denominateur);
       }
 }

public class Test{
    public static void main(String[] args) {
     try{
        Ratio a = new Ratio(2,3);
        Ratio b = new Ratio(5,7);
        a.afficher();
        b.afficher();
        Ratio c = (Ratio) (a.addition(b));
        Ratio m = (Ratio) (a.produit(b));
        System.out.println(a.egale(b));
        c.afficher();
        m.afficher();
        Ratio aa = new Ratio(1,0);
        }catch(ArithmeticException ex){
            System.out.println("Un probleme est survenu : "
            + ex.getMessage());
        }
   }
 }
```

**Correction d'exercice 3**

```
class Err extends Exception {
    public Err() {
        System.out.println("La largeur ou la hauteur
        ne doit pas etre negative");
    }
```

```
}

class Vehicule{
    private String immatriculation, marque, couleur;
    public Vehicule(String immatriculation, String marque,
    String couleur){
        this.immatriculation = immatriculation;
        this.marque = marque;
        this.couleur = couleur;
     }
    public void afficher(){
        System.out.println("L'immatriculation: " +
            immatriculation + ", marque: " + marque +
            " et couleur: " + couleur);
    }
 }

class Roue{
    private String type;
    private double largeur, hauteur;
    public Roue(String type,double largeur,double hauteur)
                            throws Err{
        if (largeur < 0 || hauteur < 0) throw new Err();
        else {
            this.type = type;
            this.largeur = largeur;
            this.hauteur = hauteur;
        }
     }
    public void afficher(){
        System.out.println("Le type: " +type+ ", largeur: "
```

```
        + largeur + " et hauteur: " + hauteur);
    }
 }

class Voiture extends Vehicule{
    private Roue a;
    public Voiture(String immatriculation, String marque,
                                String couleur, Roue a){
        super(immatriculation, marque, couleur);
        this.a = a;
    }
    public void afficher(){
        super.afficher();
        a.afficher();
    }
}

public class Test1{
    public static void main(String[] args) {
     try{
        Roue a = new Roue("R",5,3);
        Voiture b = new Voiture("a111","Passat","Gris",a);
        b.afficher();
        }catch(Err ex){
            System.out.println("Un probleme est survenu : "
                + ex);
        }
   }
 }
```

**Correction d'exercice 4**

```
import java.util.*;
public class Test{
    public static void main(String[] args){
        ArrayList<Article> A = new ArrayList<Article>();
        int i;
        try{
            Article[] tab = new Article[4];
            tab[0] = new Article("Table",800);
            tab[1] = new Article("Tableau",1500);
            tab[2] = new Fragile("TV",3000,"aa");
            tab[3] = new Fragile("PC",4000,"bb");
            for(i=0; i<tab.length-1;i++){
                tab[i].afficher();
                A.add(tab[i]);
                }
            Magasin B = new Magasin("sss",A);
            System.out.println(B.contenir(tab[3]));
            B.ajouter(tab[3]);
            System.out.println(B.contenir(tab[3]));
        }catch(ErrPrix e) {
            System.out.println(e) ;
                }
    }
}
interface Exportable{
    String paysDestination();
    double prixTransport();
}
class ErrPrix extends Exception{
    public ErrPrix(){
```

```
        System.out.println("Le prix doit etre positif");
    }
}
class Article implements Exportable {
    Scanner clavier = new Scanner(System.in);
    private static int code1 = 0;
    protected int code ;
    protected String nom;
    protected double prixHT;
   public Article(String nom, double prixHT)throws ErrPrix{
        if(prixHT <= 0) throw new ErrPrix();
        this.code = ++code1;
        this.nom = nom;
        this.prixHT = prixHT;
        }
    public String paysDestination(){
       System.out.println("Entrez le pays de destination");
        String pays = clavier.next();
        return pays;
        }
    public double prixTransport() {
        return prixHT * 0.05;
        // 5% du Prix Hors Taxes de l'article
        }
    public void afficher() {
        System.out.println("code: " + code + ", nom: "
        + nom + ", Pays:  " + paysDestination() +
        " et prix: " + (prixHT+prixTransport()));
        }
    }
class Fragile extends Article{
```

```
    String emballage;
    public Fragile(String nom, double prixHT,String
            emballage) throws ErrPrix {
        super(nom,prixHT);
        this.emballage = emballage;
        }
    public double prixTransport() {
        double a = super.prixTransport();
        return a*2;//le double du prix d'un article normale
        }
    }
class Magasin{
    private String nom;
    ArrayList<Article> A = new ArrayList<Article>();
    public Magasin(String nom,ArrayList<Article> A){
        this.nom = nom;
        this.A = A;
        }
    public void ajouter(Article B){
        A.add(B);
        }
    public Boolean contenir(Article B){
        return A.contains(B);
        }
    }
```

**Correction d'exercice 5**

```
import java.util.*;

class AgeException extends Exception{
```

```
    public AgeException(){
    System.out.println("Age ne doit pas depasser 30 ans");
    }
}

class Candidat {
    private String nom, prenom;
    private int num, age;
    public Candidat(String nom,String prenom, int num,
            int age) throws AgeException{
        if(age>30) throw new AgeException();
        this.nom = nom;
        this.prenom = prenom;
        this.num = num;
        this.age = age;
    }
    public void afficher() {
        System.out.println("Nom: " +nom + ", prenom: " +
        prenom + ", num: " +num+ " et age: " +age);
    }
  }

class Fonctionnaire extends Candidat {
    private int numSomme;
    public Fonctionnaire(String nom,String prenom, int num,
            int age, int numSomme)throws AgeException{
        super(nom, prenom, num, age);
        this.numSomme = numSomme;
        }
    public void afficher() {
        super.afficher();
```

```
        System.out.println(" Le num de
                Somme est: " +numSomme);
    }
}

public class Concours {
    public static void main(String[] args) {
    try{
        Candidat A = new Candidat("Moadi","Mohammed",1,25);
        Fonctionnaire B =new Fonctionnaire("Moad",
                    "Hanane",2,29,111);
        ArrayList<Candidat> C = new ArrayList<Candidat>();
        C.add(A);
        C.add(B);
        for(Candidat candidat: C)
                candidat.afficher();
        }catch(AgeException e){
            System.out.println(e);
        }
    }
}
```

**Correction d'exercice 6**

```
import java.util.*;
class Produit{
    private String libelle;
    private double prix;
    public Produit(String libelle, double prix){
        this.libelle = libelle;
        this.prix = prix;
```

```
    }
    public void afficher(){
        System.out.println("Libelle:"+libelle +
                " et prix: " + prix);
    }
    public double getPrix(){
        return prix;
    }
  }
class Err extends Exception{
    public Err() {
        System.out.println("La quantite doit etre
                strictement positive");
    }
 }
class Commande{
    private int numero;
    private int quantite;
    private Produit produit;
    public Commande(int numero,int quantite,Produit
                    produit) throws Err{
        if(quantite <= 0) throw new Err();
        else{
            this.numero = numero;
            this.quantite = quantite;
            this.produit = produit;
        }
    }
    public void afficher(){
        System.out.println("Numero:" + numero +
        " et quantite: " + quantite);
```

```
        produit.afficher();
    }
    public Produit getProduit(){
        return produit;
      }
   public int getQuantite(){
        return quantite;
      }
  }

class Personne{
    private String nom;
    private int age;
    public Personne(String nom, int age){
        this.nom = nom;
        this.age = age;
    }
    public void afficher(){
        System.out.println("Nom:"+ nom+ " et age: " + age);
    }
}
class Client extends Personne{
    private String nom;
    private int age;
    private Commande[] commande = new Commande[3];
    public Client(String nom,int age, Commande[] commande){
        super(nom,age);
        this.commande = commande;
    }
    public void afficher(){
        super.afficher();
```

```
        for(int i=0;i<commande.length;i++)
            commande[i].afficher();
        }
    public double payer(Commande[] commande){
        double s = 0;
        for(int i=0;i<commande.length;i++){
            s += commande[i].getProduit().getPrix() *
            commande[i].getQuantite();
           }
        return s;
       }
}

public class Test{
    public static void main(String[] args) {
        Scanner clavier = new Scanner(System.in);
        try{
            Commande [] a = new Commande[3];
            for(int i=0;i<3;i++){
                System.out.println(" Entrez le numero ");
                int b = clavier.nextInt( );
                System.out.println(" Entrez la quantite");
                int c = clavier.nextInt( );
                clavier.nextLine( );
                System.out.println(" Entrez le  libelle");
                String d = clavier.nextLine( );
                System.out.println(" Entrez le prix");
                double m = clavier.nextDouble( );
                a[i] = new Commande(b,c,new Produit(d,m));
             }
            Client c = new Client("aaa",22,a);
```

```
            c.afficher();
            System.out.println("Le cout total est: " +
                                c.payer(a));
            }catch(Err ex){
                System.out.println("Un probleme est
                survenu: " + ex);
            }
      }
}
```

**Correction d'exercice 7**

```
public class Hotel{
    public static void main(String [] args){
     try{
        ChambreDouble c = new ChambreDouble(2);
        c.afficher();
        Client a = new Client(11, "Ahmed",
                        "Mohammed", "Maroc");
        a.afficher();
        Client b=new Client(22, "Sami", "Fatima", "Maroc");
        b.afficher();
        Reservation r = new Reservation(1, a, c, 2);
        r.afficher();
        Reservation rr = new Reservation(1, b, c, 1);
        }catch(Err e) {
            System.out.println(e) ;
            }
    }
}
```

```
class Client{
    private int id;
    private String nom, prenom, pays;
    public Client(int id, String nom, String prenom,
                            String pays){
    this.id = id;
    this.nom = nom;
    this.prenom = prenom;
    this.pays = pays;
    }
    public void afficher(){
        System.out.println("Le client " + id + ": "
        + nom + ", " + prenom + " du " + pays);
    }
  }

abstract class Chambre{
    protected static int cpt;
    protected int id_chambre, num_etage, nombre_lits;
    protected boolean disponible = true;
    public Chambre(int num_etage, int nombre_lits){
        this.id_chambre = ++cpt;
        this.num_etage = num_etage;
        this.nombre_lits = nombre_lits;
    }
    public abstract void afficher();
    public void setDisponible(){
        this.disponible = !this.disponible;
    }
 }
```

```
class ChambreDouble extends Chambre{
    private final int nombre_lits = 2;
    public ChambreDouble(int num_etage){
        super(num_etage, 2);
    }
    public void afficher(){
        System.out.println("Chambre " + id_chambre+ " a " +
        nombre_lits + " lits en " + num_etage + " etage");
    }
  }

class Err extends Exception{
    public Err(){
      System.out.println("La chambre est non disponible");
    }
}

class Reservation{
    private Client client;
    private Chambre chambre;
    private int num_res, nombre_nuits;
    public Reservation(int num_res, Client client,
        Chambre chambre, int nombre_nuits) throws Err{
        if(chambre.disponible == false) throw new Err();
        this.num_res = num_res;
        this.client = client;
        this.chambre = chambre;
        this.nombre_nuits = nombre_nuits;
        chambre.setDisponible();
    }
    public void afficher(){
```

```
        System.out.println("Num reservation: " + num_res +
        " pour " + nombre_nuits + " nuits concernant ");
        client.afficher();
        chambre.afficher();
    }
}
```

**Correction d'exercice 8**

```
import java.util.*;

public abstract class Personne {
    private String nom;
    public Personne(String nom) {
    this.nom = nom;
    }
    public String getNom() {
    return nom;
    }
    public abstract void afficher();
}
class Err extends Exception{
    public Err(){
        System.out.println("La note doit
                        etre entre 0 et 20");
    }
}
public class Etudiant extends Personne{
    private String nom;
    private double[] notes = new double[3];
    Scanner clavier = new Scanner(System.in);
```

```
    public Etudiant(String nom) throws Err {
    super(nom);
    int i;
    for(i=0; i<notes.length; i++){
        System.out.println("Entrez une note ");
        notes[i] = clavier.nextDouble();
        if(notes[i]<0 || notes[i]>20) throw new Err();
        }
    }
    public double moyenne() {
    int i;
    double m = 0;
    for(i=0; i<notes.length; i++)
        m += notes[i];
    return m/notes.length;
    }

    public void afficher() {
        System.out.println("L'etudiant(e) " + getNom() +
        " obtient la moyenne " + moyenne());
    }
}
  public class GroupeEtudiant {
    private ArrayList<Etudiant> listeEtudiants = new
    ArrayList<Etudiant>();
    public int nombre() {
        return listeEtudiants.size();
    }
    public void ajouterEtudiant(Etudiant etudiant) {
        listeEtudiants.add(etudiant);
    }
```

```
    public Etudiant chercher(String nom) {
        for (Etudiant etudiant : listeEtudiants)
            if (etudiant.getNom().equals(nom))
                    return etudiant;
            return null;
    }
    public void lister() {
        System.out.println("Liste des etudiants : ");
        for (Etudiant etudiant : listeEtudiants) {
            etudiant.afficher();
        }
    }
}
public class TestGroupeEtudiants {
    public static void main(String[] args) {
        try{
        Etudiant etudiant;
        GroupeEtudiant groupe = new GroupeEtudiant();

        etudiant = new Etudiant("Mariam");
        groupe.ajouterEtudiant(etudiant);
        etudiant = new Etudiant("Hassan");
        groupe.ajouterEtudiant(etudiant);
        System.out.println("Le nombre d'etudiants dans
        la liste est:" + groupe.nombre());
        groupe.lister();
        etudiant = groupe.chercher("Mariam");
        if (etudiant != null) {
            System.out.println("Voila Mariam :  ");
            groupe.chercher("Mariam").afficher();
            }
```

```
        else System.out.println("Mariam n'est pas
                    dans la liste");
        Personne A = new Etudiant("Said");
        groupe.ajouterEtudiant((Etudiant)A);
        groupe.lister();
        }catch(Err e) {
            System.out.println(e) ;
            }
    }
}
```

**Correction d'exercice 9**

```
public class GA{
    public static void main(String[]args){
    Population p = new Population(10, 20);
    System.out.println("la population avant croisement
                    et mutation");
    p.afficher_population();
    p.croisement(p.v[0], p.v[9],15);
    p.mutation(p.v[2], 5);
    System.out.println("la population apres croisement
                    et mutation");
    p.afficher_population();
   }
}

class Population{
    int memo , taille , a , i;
    public Chromosome[] v= new Chromosome[100];
    public Population(int taille , int a){
```

```
//la population est definie par le nombre de
// chromosomes et le nombre de bits
    this.taille = taille;
    this.a = a;
    for( i=0; i<taille; i++){
        v[i]= new Chromosome(a);
            }
    }

void mutation(Chromosome v , int a){
// le a pour designer la position de mutation
    if( v.b[a-1].valeur == 0)
        v. b[a-1].valeur = 1;
    else v.b[a-1].valeur=0;
    }

void croisement( Chromosome v, Chromosome u, int r){
    for( i=r ; i<a ; i++){
    // le r ici pour designer la position de croisement
        memo = v.b[i].valeur ;
        v.b[i].valeur = u.b[i].valeur;
        u.b[i].valeur = memo;
            }
    }

void afficher_population(){
    for( int i=0; i<taille ; i++){
        v[i].afficher_chromosome();
        System.out.println();
            }
    }
```

```
}


class Chromosome{

    public Gene [] b= new Gene[100];
    int a;
    public Chromosome(int a ){
    // a est le nombre de bits associe a chaque chromosome
        this.a = a;
        for(int i=0; i<a; i++){
            b[i]= new Gene();
                }
            }

    void afficher_chromosome(){
        for(int i=0; i<a; i++){
            System.out.print(b[i].valeur);
            }
        }
    }

class Gene{
    int valeur;
    public Gene(){
        valeur = (int) (Math.random() + 0.5) ;
    }
}
```

**Correction d'exercice 10**

```
class Err extends Exception{
    public Err(){
        System.out.println("Direction incorrecte");
    }
}
abstract class Robot{
    private String nom;
    private int x;
    private int y;
    private String direction;
    public Robot(String nom, int x, int y, String
                direction) throws Err{
        this.nom = nom;
        this.x = x;
        this.y = y;
        if (direction.equals("Nord") ||
            direction.equals("Sud")||
            direction.equals("Ouest") ||
            direction.equals("Est"))
                this.direction = direction;
        else throw new Err();
        }
    /**
        * avance d'un pas
    */
    public void avance(){
        if (direction.equals("Nord")) y++;
        else if (direction.equals("Est")) x++;
        else if (direction.equals("Sud")) y--;
        else x--; // (direction.equals("Ouest"))
```

```
        }
    /**
        * tourne a droite de 90
    */
    public void droite(){
      if (direction.equals("Nord")) direction = "Est";
      else if (direction.equals("Est")) direction = "Sud";
      else if (direction.equals("Sud")) direction ="Ouest";
      else direction = "Nord";
      // (direction.equals("Ouest"))
        }
    /**
        * affiche l'etat du robot
    */
    public void afficher(){
        System.out.println("nom : " + nom + ", position: ("
        + x + "," + y +")" + " et direction: " +direction);
        }
    public abstract void gauche();
    public abstract void demiTour();
    public abstract void avance(int pas);
    }

class RobotNG extends Robot{
    public RobotNG(String nom, int x, int y, String
                direction) throws   Err{
        super(nom, x, y, direction);
        }
    /*
         * avance de plusieurs pas
    */
```

```
    public void avance(int pas){
        for (int i = 0 ; i < pas ; i++) {
             avance();
            }
        }

    /**
        * tourne a gauche de 90
    */
    public void gauche(){
        droite();
        droite();
        droite();
        }
    /**
        * fait demi-tour
    */
    public void demiTour(){
        droite();
        droite();
        }
    }

public class RobotExam{
    public static void main(String[] args){
     try{
        Robot[] tableau = new RobotNG[3] ;
        /*tableau[0] = new Robot("a",2, 3, "Nord"); est
        impossible car la classe robot est abstraite
        */
        tableau[0] = new RobotNG("a",2, 3, "Nord");
```

```
        tableau[0].avance();
        tableau[0].droite();
        tableau[1] = new RobotNG("b",22, 13, "Ouest");
        tableau[1].avance(3);
        tableau[1].droite();
        tableau[2] = new RobotNG("c",12, 31, "Sud");
        tableau[2].avance(7);
        tableau[2].gauche();
        tableau[2].demiTour();
        for (Robot r : tableau) {
                if (r != null) r.afficher();
               }
        }catch(Exception e){
                e.toString();
            }
    }
}
```