1) Cours

1-1) comparaison tableau classique et ArrayList

les tableaux classique sont de taille fixe, définie au début de l'exécution du programme. Ils n'offrent que de basiques possibilités pour la gestion de « groupement » de données. Ils ne peuvent stocker que des données de types basiques.
Les arrayList sont de taille extensible. On peut y rajouter ou supprimer des données au fur et à mesure de l'exécution du programme, changeant ainsi la taille de la structure. Les ArrayList stockent des données de type dits évolués (des objets). De plus, ils font partis du Java Collection Framework, et offrent donc un ensemble de possibilités qu'on a pas avec les tableaux classiques.

1-2)

a) il s'agit d'une instruction
b) les opérateurs sont + et *
c) les opérandes de * sont 2 et x
d) si x vaut trois, cette instruction affichera : Resultat = 7
e) si on supprime les parenthèses, le résultat reste le même car la multiplication est prioritaire sur l'addition

2) Erreurs

2-1)

a) erreur 1, ligne 02 : la méthode main est mal déclarée.
Correction : public static void main(String [] args)
b) erreur 2, ligne 03 : chiffre est déclaré int mais reçoit un double.
Correction : int chiffre = 1 ;
c) erreur 3, ligne 05 : chiffre est mal orthographié.
Correction : écrire chiffre (comme à la déclaration)
d) erreur 4, ligne 07 : il manque un « ; ».
Correction : rajouter le « ; » à la fin de l'instruction
e) erreur5, ligne 09 : la condition du if doit être mise dans un bloc de parenthèses pour être interprété par comme une seule et même condition.
Correction : if((chiffre < 1) || (chiffre > 9))
f) erreur6, ligne 23 : erreur d'accolade. En effet, une accolade ouvrante manque à la ligne 16 pour marquer l'ouverture du bloc for dont la fermeture est marqué par l'accolade en ligne 21.
Correction : rajouter une accolade ouvrante for(int i=1;i<10;i++){

2-1) Pour corriger cette dernière erreur, il faut correctement déclarer la fonction main qui le point d'entrée (donc la méthode principale) de tout programme java.

a) erreur 1, ligne 02 : la méthode main est mal déclarée.
Correction : public static void main(String [] args)
D'autres parts, même si java considère public les classes sans modificateurs d'accès, les bonnes pratiques préconisent de faire cette précision. Il faudrait donc rajouter le mot clé public devant le mot class en ligne 01.

3) Algorithmique

3-1)

```
Entrées :
     n : un entier
Sorties :
     res : le nombre premier trouvé
Variables locales :
     trouve : un booléen qui passe à vrai quand un nombre premier
              est trouvé
     compteur : un entier, compteur de diviseur
     i : un entier
Algorithme :
     trouve = faux
     tant que trouve == faux
          compteur = 0
          pour i allant de 2 à n
               si n modulo i == 0
                    compteur = compteur + 1
          si compteur == 1
               trouve = true
               res = n
          sinon
               n = n + 1
```

3-2)

```
public static int compteur(String texte){
     int nbMots = 0 ;
     int pos = 0 ;
     while(pos < texte.length){
          while(pos < texte.length){
               pos++ ;
          }
     if(pos<texte.length){
          nbMots++ ;
          while(pos < texte.length){
               pos++ ;
          }
     }
     }
     return nbMots ;
}
```

5) Exercices

5-1)

```
public static int [] fraction(int num, int den){
     int [] tab = new int[2] ;
     if (den<=0) {
          System.out.prinln(‘’dénominateur interdit’’)
     } else {
          tab[0] = num ;
          tab[1] = den ;
     }
     return tab ;
```

```
}
```

5-2)

```
public static int [] simplification(int [] fraction){
     int diviseur = pgdc(fraction[0], fraction[1]) ;
     fraction[0] = fraction[0] / diviseur ;
     fraction[1] = fraction[1] / diviseur ;
     return fraction ;
}
```

5-3)

```
public static boolean cherche(int [] f, ArrayList<int []> ens){
     int i = 0 ;
     boolean trouve = false ;
     f = simplification(f) ;
     while(!trouve && i<ens.size()){
          if(f[0] == ens.get(i)[0] && f[0] == ens.get(i)[0]){
               trouve = true ;
          }
     }
     return trouve ;
}
```

5-4)

```
public static void ajouter(int [] f, ArrayList<int []> E){
     if( !cherche(f, E){
          E.add(simplification(f)) ;
     }
}
```

5-5)

```
public static ArrayList<int []> fusion(ArrayList<int []> ens1,
                                       ArrayList<int []> ens2) {
     ArrayList<int []> resultat = new ArrayList<int []>() ;
     resultat.addAll(ens1) ;
     resultat.addAll(ens2) ;
     return resultat ;
}
```