Correction des exercices de travaux dirigés

Exercice 1 : notes

1-a) Définition du type Etudiant

```
class Etudiant {
    double math;
    double ang;
    String nom;
}
```

1-b) Définition de la procédure de saisie

```
void saisir(Etudiant etud) {
    etud.nom = readString( "nom : " );
    etud.math = readDouble( "math : " );
    etud.ang = readDouble( "anglais : " );
}
```

1-c) Définition de la fonction calculant l'avis

```
String calculer_avis(Etudiant etud) {
   double moy;
   String avis;
   moy = (etud.math+etud.ang)/2;
   if (moy<10) {
      avis = "refuse";
   }
   else if (moy<12) {
      avis = "passable";
   }
   else if (moy<14) {
      avis = "assez bien";
   }
   else if (moy<16) {
      avis = "bien";
   }
   else {
      avis = "tres bien";
   }
   return avis;
}
```

2) Programme principal gérant un étudiant

```
void main() {
   Etudiant etud = new Etudiant();
   String avis;
   saisir(etud);
   avis = calculer_avis(etud);
   println( "avis : " + avis );
}
```

3) Programme principal gérant plusieurs étudiants

```
void main() {
   int nbEtud;
   Etudiant etud = new Etudiant();
   String avis;
   int k;
   int nbRefuse;
   nbEtud = readInt( "nbEtud : " );
   nbRefuse = 0;
   for(k=1; k<=nbEtud; k=k+1) {
      saisir(etud);
      avis = calculer_avis(etud);
      println( "avis : " + avis );
      if (equal(avis,"refuse")) {
         nbRefuse = nbRefuse+1;
      }
   }
   println( "nbRefuse : " + nbRefuse );
}
```

Exercice 2 : compétition de saut à skis

```
class Skieur {
   double [] notes;
   double noteFinale;
   String nom;
}
```

1-a) Définition de la procédure de saisie

```
void saisir(Skieur sk) {
   sk.nom = readString( "nom : " );
   sk.notes = new double[5];
   int k;
   for(k=1; k<=5; k=k+1) {
      sk.notes[k-1] = readDouble( "note " + (k) + " : " );
   }
}
```

1-b) Définition de la procédure calculant la note finale

```
void calculerNoteFinale(Skieur sk) {
   int k;
   double somme;
   double mini;
   double maxi;
   somme = sk.notes[0];
   mini = sk.notes[0];
   maxi = sk.notes[0];
   for(k=2; k<=5; k=k+1) {
      somme = somme + sk.notes[k-1];
      if (sk.notes[k-1] < mini) {
         mini = sk.notes[k-1];
      }
      if (sk.notes[k-1] > maxi) {
         maxi = sk.notes[k-1];
      }
   }
   sk.noteFinale = somme - mini - maxi;
}
```

1-c) Programme principal gérant un skieur

```
void main() {
   Skieur sk = new Skieur();
   saisir(sk);
   calculerNoteFinale(sk);
   println( sk.nom + " a pour note finale " + sk.noteFinale );
}
```

2) Programme principal gérant plusieurs skieurs et tri

```
void main() {
   int nbSkieur;
   Skieur[] tab;
   int i;
   // saisie
   nbSkieur = readInt( "nombre de skieurs : " );
   tab = new Skieur[nbSkieur];
   for(i=1; i<=nbSkieur; i=i+1) {
      tab[i-1] = new Skieur();
      saisir(tab[i-1]);
   }
   // calcul des notes finales
   for(i=1; i<=nbSkieur; i=i+1) {
      calculerNoteFinale(tab[i-1]);
   }
   //  tri par note finale décroissante
   Skieur copie;
   int k;
   for(i=1; i<=nbSkieur-1; i=i+1) {
      for(k=i+1; k<=nbSkieur; k=k+1) {
         if (tab[i-1].noteFinale < tab[k-1].noteFinale) {
            copie = tab[i-1];
            tab[i-1] = tab[k-1];
            tab[k-1] = copie;
         }
      }
   }
   // affichage du classement
   for(i=1; i<=nbSkieur; i=i+1) {
      println( tab[i-1].nom + " a pour note " + tab[i-1].noteFinale );
   }
}
```