SUGGESTIONS POUR LA PARTIE B2.1 :

Dans la classe `Calculette`, vous pourriez…

- Déclarer la constante suivante :
```
public static final String[] opBinaire = new String[]{"divise", "exposant", "moins",
"multiplie", "plus"};
```
- Déclarer et coder les procédures suivantes :

```
public static float calcul(float f1, float f2, String op)
        throws EDivideByZero, EBadExponentation, EUnknownOP {
    //{op représente une opérateur binaire} =>
    //  {résultat = réel résultat de l'opération f1 op f2 si elle est valide
    //    - l'exception EDivideByZero est levée si f2=0
    //      et op est l'opérateur de division
    //    - l'exception EBadExponentation est levée si f1<0 et f2<1
    //      et op est l'opérateur d'exponentiation
    //    - l'exception EUnknownOP est levée si op n'est pas un opérateur connu}
```

```
public static boolean estUnOpBin(String op) {
    // {} => {résultat = vrai si op est un opérateur binaire}
```

```
public static boolean estUnFloat(String s) {
    //{} => {résultat = vrai si s représente un float}
```

```
public static float resExp(String exp, PileFloat pCalc)
        throws EPilePleine, EPileVide, EDivideByZero, EBadExponentation, EUnknownOP {
    //{exp représente une séquence de calcul RPN (ex : 3 ENTER 2 plus 5 multiplie}
    //   => {résultat = valeur résultant du calcul s'il est possible, sinon
    //       l'exception levée est traitée et un message approprié est affiché}
```

Algorithme proposé pour cette fonction :

1 – Utiliser la méthode `split` de la classe `String` pour construire un tableau de chaînes à partir de la chaîne `exp` représentant la séquence à traiter
Exemple : *Si* `exp` = "5 ENTER 2 plus" *alors* `exp.split(" ")` = ["5", "ENTER", "2", "plus"]

2 – Faire une boucle de parcours du tableau de chaînes obtenu à partir de `exp`
Dans cette boucle…
* Identifier la nature de l'élément courant (*nombre*, *opérateur*, *ENTER*, *inconnu*…)
* Effectuer les actions nécessaires selon la nature de cet élément (*mémorisation pour calcul ultérieur, empilage, dépilage, levée d'exception, calcul…*)

3 – Retourner le résultat final