Rappel : ne pas oublier de s'inscrire sur Didel.

Remarque générale : dans les TP il est toujours attendu que vous testiez chacune de vos méthodes sur des exemples.

Exercice 1 (Introduction à jGRASP)

1. Lancez jGRASP avec la commande `jgrasp` dans un terminal.
2. Copiez le code suivant et sauvez-le dans un fichier `EssaiJGrasp.java` :

```
class EssaiJGrasp {
   public static void main(String[] args) {
      int v = 1;
      v = -2;
      int[] tab = new int[4];
      tab[0] = 6;
      tab[1] = -1;
      tab[2] = 10;
      tab[3] = 14;
      System.out.println("Test fini!");
   }
}
```

3. Affichez les numéro de ligne avec `Ctrl+L`.
4. Compilez (bouton ✚ ou `Ctrl+B`) puis exécutez (bouton 🏃 ou `Ctrl+R`) le programme. Ces étapes reviendraient à taper dans un terminal `javac EssaiJGrasp.java` puis `java EssaiJGrasp`.
5. Un des avantages de jGRASP est que l'on peut visualiser l'évolution des objets en mémoire. Au lieu d'exécuter comme précédemment, commencez par faire un clic gauche dans la marge à côté de la ligne `int v = 1;` pour faire apparaître un point rouge. Faites de même à côté des lignes suivantes jusqu'à `tab[3] = 14;` puis lancez le canevas (bouton 📺). Une fenêtre secondaire s'ouvre et va nous servir à visualiser les objets.
6. Avancez d'une étape (bouton ⬇ ou `Alt+↓`). Dans la partie gauche de la fenêtre principale cliquez sur le symbole ◣ à côté de la variable `v` et glissez-le dans la fenêtre secondaire.
7. Avancez encore d'une étape et observez le changement de la représentation de `v`.
8. Avancez d'une étape, puis faites de même glisser le symbole ■ correspondant à la variable `tab` dans le canevas. Puis continuez pas à pas jusqu'à la fin en observant le canevas.

Exercice 2 Soit le code suivant, qui permet de représenter un fruit par son nom et son poids :

```
class Fruit {
    String nom; // le nom du fruit
    double p; // le poids du fruit en grammes

    public Fruit() {
        this.nom = "Fruit-sans-nom";
        this.p = 0;
    }
    /* A COMPLETER ... */
}
```

Le constructeur `Fruit()` ci-dessus ne prend aucun argument et crée un fruit se nommant "Fruit-sans-nom" d'un poids de 0 gramme.

1. On souhaiterait afficher une description du fruit. Dans une classe principale `ClassePrincipale` (indépendante de la première classe) écrivez une fonction d'en-tête : `static void affiche(Fruit f)` qui prend un fruit en argument (et ne retourne rien), et qui affiche un message sous la forme "Ce fruit est un(e) xxx et pèse yyy gramme(s)." (où "xxx" est le nom du fruit et "yyy" son poids).
2. Testez votre constructeur en créant dans un `main` une instance de la classe `Fruit`, puis en l'affichant.
3. Ajoutez à la classe `Fruit` un constructeur qui prend en arguments un nom (de type `String`) et un poids (de type `double`) et les affecte à l'attribut correspondant dans la classe.
4. Testez ce nouveau constructeur en créant une instance correspondant à un citron (que vous pourrez nommer par exemple "citron" ou "mon petit citron") de 100g.
5. Ecrivez dans la classe `ClassePrincipale` une fonction `static Fruit ajout(Fruit f1, Fruit f2)` qui prend deux fruits et retourne un `Fruit`. Le fruit retourné est créé dans le corps de la fonction et a les caractéristiques suivantes : son poids est la somme des poids des deux fruits, et son nom est la concaténation des deux noms séparés par le symbole "+".
6. Testez cette fonction en ajoutant un melon de 400g au citron précédent puis affichez le résultat.
7. Il n'est pas très logique qu'une somme de fruits soit encore un fruit. Construisez une classe `Panier` qui contient comme seul attribut un tableau de `Fruits`.
8. Nous allons ajouter plusieurs constructeurs pour cette classe :
   - `Panier()` qui ne prend aucun argument et crée un panier vide (c'est à dire que le panier ainsi créé ne contient rien, mais il n'est pas `null`) ;
   - `Panier(Panier p)` qui prend en argument un autre panier, `p`, et le copie. Regardez ce qui se passe avec jGRASP (si vous ne copiez pas soigneusement le tableau de fruits de `p`, c'est seulement la référence de ce tableau qui est copiée) ;
   - `Panier(Fruit[] f)` qui prend en argument un tableau de fruits et construit le panier contenant ce tableau.
   - `Panier(Fruit f, Panier p)` qui prend en argument un fruit `f` et un panier `p` et qui construit un nouveau panier dont l'ensemble des fruits est celui de `p` plus le fruit `f`. Attention, si le panier `p` est `null`, le nouveau panier ne contient que le fruit `f`.
9. Ecrivez une fonction `static Panier ajout(Fruit f, Panier p)` qui retourne le panier comprenant les fruits du panier `p` plus le fruit `f` (sans oublier de traiter le cas où le panier `p` donné en argument est `null`).

Exercice 3 *[si le temps le permet]*

Soit le début de classe suivant :

```
class Etudiant {
    String nom; // le nom de l'etudiant
    String prenom; // le prenom
    int num; // numero d'etudiant
    double note; // la note de l'etudiant (sur 20)
    /* A COMPLETER ... */
}
```

Celle-ci permet de stocker quelques informations relatives à un étudiant.

1. Ajoutez à cette classe un constructeur prenant en argument un nom, un prénom, un numéro d'étudiant ainsi qu'une note et les affectant aux attributs concernés (on suppose que l'utilisateur donne toujours une notre entre 0 et 20). Le constructeur aura ainsi comme en-tête : `public Etudiant(String nomEtu, String prenomEtu, int numEtu, double noteEtu)`.
2. Dans cette classe principale :
   (a) écrivez une fonction `static void affiche(Etudiant etu)` pour afficher le contenu de l'objet au format "Nom Prénom (Numéro d'étudiant) : Note" ;

(b) créez un `main` pour tester votre classe `Etudiant` en construisant un exemple puis en l'affichant *(vous testerez systématiquement chaque nouvelle fonction)* ;

(c) écrivez une fonction `static char[] initiales(Etudiant etu)` qui prend en argument un `Etudiant` et renvoie un tableau contenant exactement deux caractères (de type `char`) : le premier correspond à l'initiale du prénom et le deuxième à l'initiale du nom ;[1] vous pourrez utiliser la méthode `charAt` de la classe `String`, qui prend en argument la position de la lettre qu'on souhaite obtenir dans la chaîne ;

(d) ajoutez une méthode `passage` qui prend en argument un étudiant et renvoie le booléen disant si l'étudiant peut passer (c'est-à-dire si sa note est supérieure ou égale à 10) ;

(e) écrivez enfin une fonction qui donne la mention de l'étudiant : "Très bien", "Bien", "Assez bien", "Passable" ou "Redouble" si sa note est respectivement dans l'intervalle $[16, 20]$, $[14, 16[$, $[12, 14[$, $[10, 12[$ ou $[0, 10[$. Si la note n'est dans aucun de ces intervalles, la fonction retourne "Note invalide". Elle prend donc un argument de type `Etudiant` et renvoie une chaîne de caractères (type `String`). On pourra utiliser des `if ... else ...` imbriqués.

1. Un `return` ne peut renvoyer qu'un seul objet. Il existe des façons élégantes en Java pour contourner cette difficulté lorsqu'on veut qu'une méthode retourne plusieurs objets, mais nous nous limitons ici à l'utilisation d'un petit tableau.