**Seul document autorisé : une feuille A4 recto-verso manuscrite (non photocopiée). Aucune machine.**
**Le barème est indicatif. Une question peut toujours être traitée en utilisant les précédentes (traitées ou non).**
**Les morceaux de code Java devront être clairement présentés, indentés et commentés.**
**L'utilisation de collections de l'API Java (en particulier `LinkedList` et `ArrayList`) est interdite.**

— Tous les attributs d'instance de toutes les classes introduites doivent être privés.
— Il est possible d'écrire des méthodes non spécifiquement demandées si c'est plus pratique, en spécifiant pour chaque méthode dans quelle classe elle se trouve.

**Exercice 1.** (1 point) *Question de cours (indépendante de la suite de l'énoncé).* Expliquer brièvement (on ne demande pas nécessairement de code) comment on a encodé en cours un arbre d'arité quelconque (c'est-à-dire un arbre dont les nœuds n'ont pas tous le même nombre de fils) par un arbre binaire.

Dans cet énoncé on se propose d'introduire la structure de données et quelques méthodes pour représenter un arbre de Pythagore (voir figure 1 gauche) construit de manière récursive, comme illustré dans la partie droite de la figure 1.

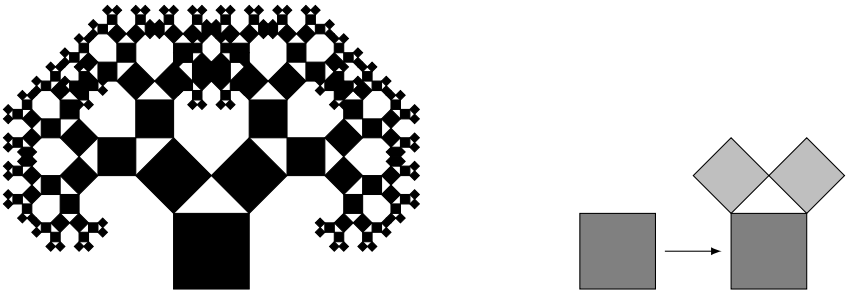


FIGURE 1 – Arbre de Pythagore de profondeur 8 et passage d'un niveau au suivant.

# 1 Carrés

On veut écrire une classe `Carre` représentant un carré coloré du plan. Ce carré sera représenté par son centre de type `java.awt.Point`, un angle de type `double` donnant l'orientation du carré, un `double` donnant la longueur de ses côtés et une couleur de type `java.awt.Color`. Par ailleurs chaque carré doit avoir un identifiant entier unique.

L'interface de la classe `Carre` est la suivante :

```
/** renvoie la couleur du carre */
Color getCouleur();
/** fixe la couleur du carre */
void setCouleur();
/** renvoie l'identifiant du carre */
int id();
/** dessine le carre -- non demandee dans l'exercice 2 */
void dessine();
/** renvoie un tableau contenant les deux carres suivants dans la construction,
on specifie leurs couleurs en parametre -- non demandee dans l'exercice 2 */
Carre[] suivants(Color c0, Color c1);
/** renvoie un tableau contenant les deux carres suivants dans la construction,
avec la meme couleur que le carre d'origine */
Carre[] suivants();
```

La classe `Carre` contient également un constructeur qui renvoie un carré noir centré en $(0, 0)$, d'angle $90$ et de côté $100$ (qui est le carré de départ pour l'arbre de Pythagore).

**Exercice 2.** (2 points) Écrire la classe `Carre`, sauf les méthodes `dessine()` et `suivants(Color c0, Color c1)` (penser à écrire la méthode `suivants()`). On rappelle que le noir correspond à l'attribut constant `BLACK` de la classe `java.awt.Color`.

# 2 Arbres

On représente un arbre de Pythagore par un arbre binaire, chaque nœud de l'arbre représentant un carré. La classe `ArbrePy` référence la racine de cet arbre et la classe `NoeudPy` représente ses nœuds. On considère qu'un arbre est de hauteur $0$ s'il contient un unique nœud (sa racine).

**Exercice 3.** (0,5 point) Donner les attributs des classes `ArbrePy` et `NoeudPy`.

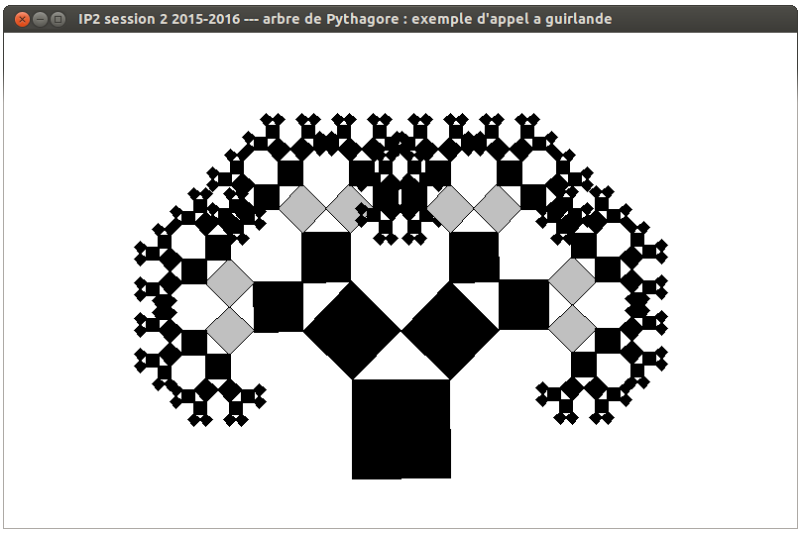
**Exercice 4.** (2 points) Écrire un constructeur pour la classe **`ArbrePy`** qui prend en paramètre la hauteur de l'arbre à créer.

L'interface de la classe **`ArbrePy`** est la suivante :

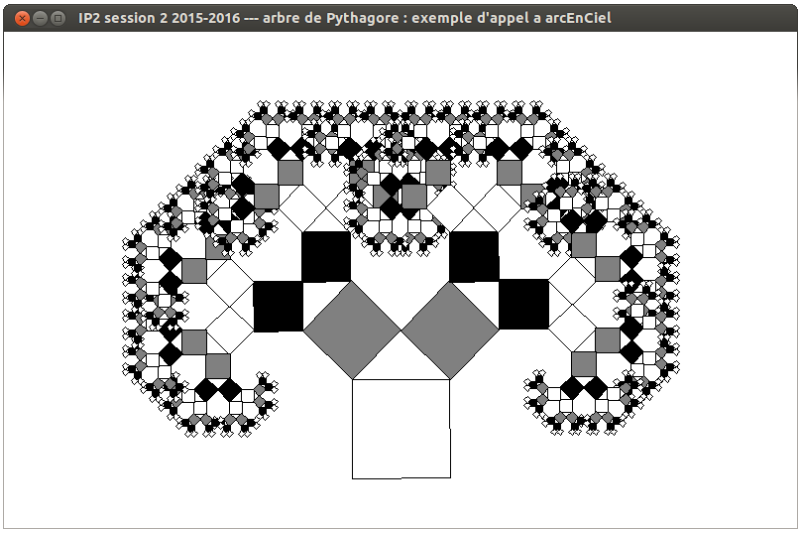
```
/** dessine (ou redessine) tous les carres de l'arbre */
void dessine();
/** definit les couleurs des carres de l'arbre en alternant les couleurs par niveau*/
void arcEnCiel(Color[] c);
/** place une guirlande de couleur c a la profondeur n */
void guirlande(Color c, int n);
/** renvoie le nombre de carres noirs */
int nbCarresNoirs();
/** laisse les carres dont l'identifiant est divisible par le parametre k inchanges
et passe la couleur des autres a noir */
void loupiottes(int k);
/** affiche la liste des couleurs de l'arbre avec un parcours en largeur */
void parcoursEnLargeur();
```

**Exercice 5.** (3 points) Écrire les méthodes **`dessine`** et **`nbCarresNoirs`**.

**Exercice 6.** (2 points) Écrire la méthode **`guirlande`** : les couleurs de tous les carrés au niveau **`n`** passent à **`c`**, les autres restent inchangées. Un exemple où tous les carrés sont noirs avant l'appel à la méthode :




**Exercice 7.** (2 points) Écrire la méthode **`arcEnCiel`** : le carré à la racine de l'arbre prend comme couleur la première couleur du tableau, les carrés du niveau suivant la couleur suivante, *etc*. Si le tableau de couleurs a moins de cases que le nombre de niveaux de l'arbre, après avoir attribué la dernière couleur on repart sur la première. Un exemple :




**Exercice 8.** (2 points) Écrire la méthode **`loupiottes`**.

On veut maintenant faire un parcours en largeur de l'arbre (c'est-à-dire par niveau). Pour cela on utilise une file (premier entré/premier sorti) de nœuds.

**Exercice 9.** (4 points) Mettre en place les classes nécessaires pour gérer une file de **`NoeudPy`**.

**Exercice 10.** (2,5 points) Écrire la méthode **`parcoursEnLargeur`**.