TP4: Récursivité sur les entiers

— **Comme son nom l'indique, ce TP a pour but de vous faire travailler sur des fonctions récursives. L'usage de toute boucle `for` ou `while` est donc interdit durant ce TP.**

— **Les exercices marqués d'une étoile (*) sont à traiter en priorité en TP. Les autres sont à traiter en TP s'il vous reste du temps, ou bien en temps personnel.**

1 Préliminaires

Configuration En vous aidant de la Notice ScalaIDE :

1. Lancez l'éditeur ScalaIDE (Eclipse 4.7 Scala)
2. Importez le projet ScalaIDE `TP4.zip` disponible dans Moodle.

Auto-évaluation des TP Pour vous aider à auto-évaluer votre travail, nous vous fournissons quelques tests unitaires. La Notice ScalaIDE décrit la procédure à suivre pour exécuter ces tests. Aucun test n'est fourni ni attendu pour les exercices portant sur les images.

2 Récursivité en images : package `fr.istic.si2.tp4.exoImages`

Exercice 1 (Immeuble*). *Programmez une fonction récursive* `immeuble`*, qui étant donné un entier positif ou nul n, produit une image symbolisant un immeuble de n étages, couvert d'un toit. Pour ce faire, utilisez les images* `toit` *et* `etage` *fournies. Cette fonction vous a été présentée en cours. Testez votre fonction sur quelques exemples, en utilisant les fonctions* `draw`*.*

Exercice 2 (Collier de perles*). *Programmez une fonction récursive* `collierPerles` *qui, étant donné un entier positif ou nul n, produit une image d'un collier de perles horizontal, constituée de n répétitions de l'image* `perle` *fournie. Ci-dessous, on représente l'image renvoyée par* `collierPerles(7)`*. Testez votre fonction sur quelques exemples, en utilisant les fonctions* `draw`*.*

Exercice 3 (Anneaux*). *Programmez une fonction récursive* `anneaux`*, qui étant donné un entier positif ou nul n, produit une image constituée de n anneaux concentriques de plus en plus grands. Pour ce faire, utilisez la fonction* `anneau` *fournie. Ci-dessous, on représente l'image renvoyée par* `anneaux(5)`*. Testez votre fonction sur quelques exemples, en utilisant les fonctions* `draw`*.*

Exercice 4 (Guirlande). *Programmez une fonction récursive* `guirlande`*, qui étant donné un entier positif ou nul n, produit une image de guirlande horizontale composée de $2 * n$ boules de deux couleurs, disposées en alternant les deux couleurs. Pour ce faire, utilisez* `boule_rouge` *et* `boule_noire`*.*

Testez votre fonction sur quelques exemples, en utilisant les fonctions `draw`*.*

Exercice 5 (GuirlandeBis – Difficile)**.** *Programmez une fonction récursive* `guirlandeBis`*, qui étant donné un entier positif ou nul n, produit une image de guirlande horizontale composée de n boules de deux couleurs, disposées en alternant les deux couleurs, en commençant, à gauche, par une boule rouge. Attention, à la longueur de la guirlande (elle doit être composée de n boules exactement). Pour ce faire, utilisez* `boule_rouge` *et* `boule_noire`*.*

Testez votre fonction sur quelques exemples, en utilisant les fonctions `draw`*.*

3 Répéter et calculer récursivement : package `fr.istic.si2.tp4.exosCalcul`

Exercice 6 (*)**.** *Programmez une fonction récursive* `copie`*, qui étant donné un entier positif ou nul n et une chaîne s, renvoie la chaîne composée de n fois la chaîne s, en utilisant le tiret comme séparateur.*
Par exemple, `copie(1, "blah")` *renvoie la chaîne* `"blah"` *et* `copie(3, "blah")` *renvoie la chaîne* `"blah-blah-blah"`*. Attention, il n'y a de tiret ni au début ni à la fin de la chaîne produite en résultat.*

Testez votre fonction sur quelques exemples, en utilisant des `println`*.*

Exercice 7 (*)**.** *Programmez une fonction récursive* `repeat`*, qui étant donné un entier positif ou nul n et un caractère c, affiche n fois le caractère c dans la console, avec un retour à la ligne entre chaque caractère.*

Testez votre fonction sur quelques exemples.

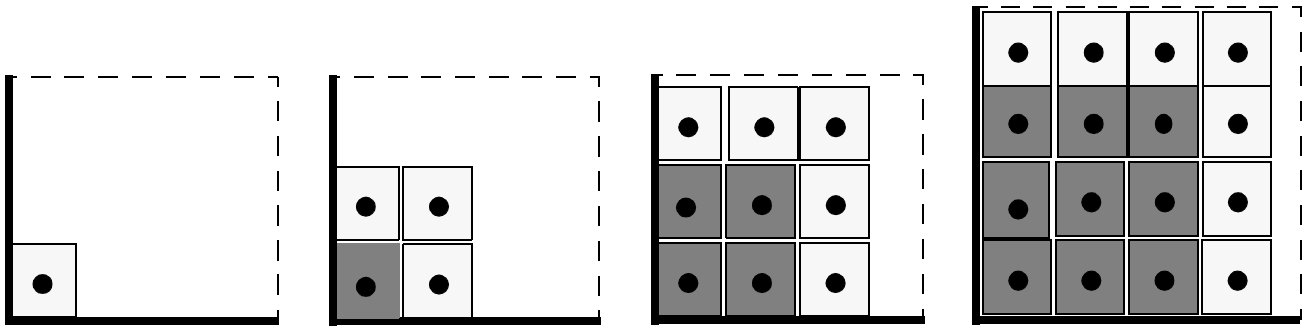
Exercice 8. *Programmez une fonction récursive* `countDown`*, qui étant donné un entier positif ou nul n, affiche dans la console le décompte en partant de n jusqu'à 1, en finissant par* `"Time␣OUT␣!"`*, avec un retour à la ligne entre chaque nombre.*

Testez votre fonction sur quelques exemples.

Exercice 9 (Les nombres carrés *)**.** *Les pythagoriciens définissaient les* nombres carrés *géométriquement :*

1. *le premier nombre carré est formé d'un élément de base placé sur une grille ;*
2. *le nombre carré suivant est obtenu en entourant le précédent d'autant d'éléments de base que nécessaire pour former un nouveau carré.*

La répétition de ce procédé produit les nombres carrés successivement, selon le schéma suivant :



Le nombre d'éléments de base utilisés pour construire de telles figures, c'est-à-dire 1, 4, 9, 16. . ., étaient donc appelés nombres carrés.

Programmez une fonction récursive `nombreCarre`*, qui étant donné un entier strictement positif n, renvoie l'entier* n^2*,* ***sans faire usage de la multiplication****. Indication : essayez d'exprimer* n^2 *en fonction de* $(n-1)^2$ *en vous aidant des schémas ci-dessus.*

Testez votre fonction sur quelques exemples, en affichant ses résultats avec des `println`*.*

4 Fractales : package `fr.istic.si2.tp4.fractales`

Ces exercices sont en bonus. Avant de vous y intéresser, veuillez vous assurer que vous avez réalisé les exercices précédents de manière exemplaire : aucune erreur et aucun warning Scalastyle ne doit rester, et tous les tests unitaires doivent réussir.

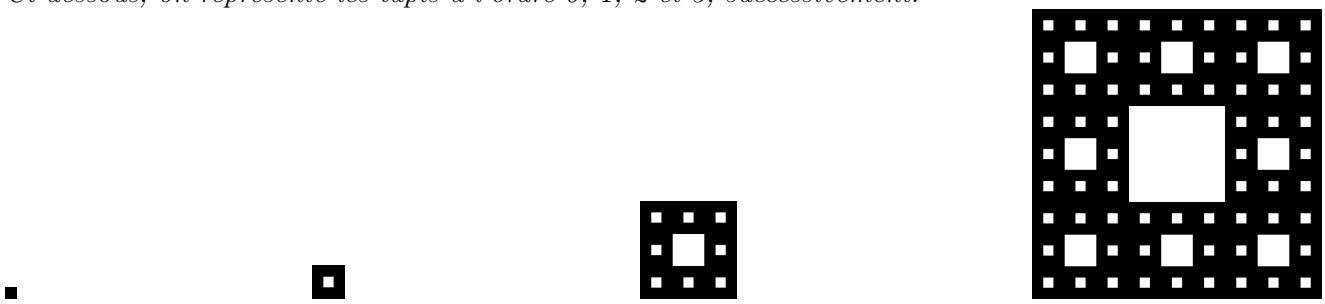
Pour afficher les images produites avec Scribble, on vous rappelle l'existence des fonctions `draw`, dont une qui permet de nommer la fenêtre graphique qui s'ouvre à l'affichage (voir documentation Scribble).

Exercice 10 (Triangle de Sierpinski)**.** *Programmez une fonction récursive* `sierpinski` *qui, étant donné un entier positif ou nul n, construit une image représentant le triangle de Sierpinski à l'ordre n.*

Ci-dessous, on représente les triangles à l'ordre 0, 1, 2 et 3, successivement.

Exercice 11 (Tapis de Sierpinski)**.** *Programmez une fonction récursive* `tapis` *qui, étant donné un entier positif ou nul n, construit une image représentant le tapis de Sierpinski à l'ordre n.*

Ci-dessous, on représente les tapis à l'ordre 0, 1, 2 et 3, successivement.



Exercice 12 (Rosace – Difficile)**.** *Programmez une fonction récursive* `rosace` *qui, étant donné un entier positif ou nul n, génère une image de rosace à l'ordre n, suivant le motif suivant (on tolère un motif similaire).*

Ci-dessous, on représente la rosace à l'ordre 0, 1, 5 et 10.