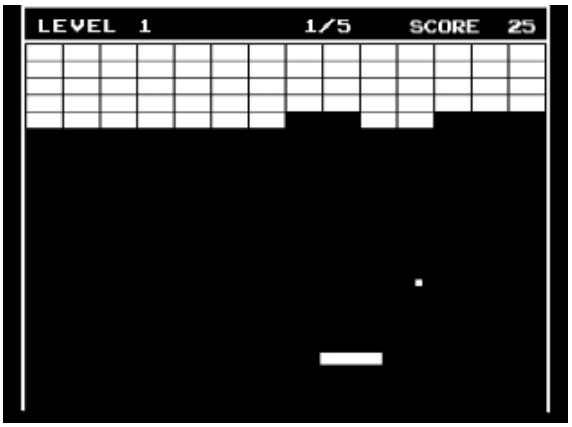
TP 2 : Plan de travail, navigation et édition avancée

Ce TP a pour but de vous familiariser avec les différentes opérations introduites en cours concernant d'une part la gestion de ressources et d'autre part l'édition et la navigation dans du code source Java.

Tout le TP consiste à écrire un jeu de « casse-briques », cf. Wikipédia :

`http://fr.wikipedia.org/wiki/Breakout`
`http://fr.wikipedia.org/wiki/Arkanoid`



Comme la structure de ce jeu est assez proche du jeu *Pong* du TP 1, le code source du jeu *Pong* du TP 1 va en grande partie servir à nouveau.

Dans toute la suite, s'efforcer de garder un code bien présenté à l'aide des commandes *Correct Indentation* et *Format*.

Création du projet

À partir de l'ancien projet *Pong*, un nouveau projet *Breakout* va être créé avec quelques aménagements (renommage de package, de classes et de champs).

Exercice 1 :
- Récupérer la correction du TP 1, disponible sous forme d'archive à l'adresse suivante :
  `http://www.di.ens.fr/~labatut/ED6/tp-1/Pong-correction.zip`
- Importer le projet depuis l'archive dans votre espace de travail.
- Vérifier que le projet fonctionne.
- Créer un nouveau projet vide *Breakout* de type *Java Project*.
- Rajouter un nouveau package `breakout` dans ce projet.

Exercice 2 :
- Ouvrir la vue *Navigator*, et copier toutes les sources du *Pong-correction* (utiliser éventuellement des filtres...) dans le répertoire `breakout` du projet *Breakout*.
- Comparer les fichiers correspondants de *Pong-correction* et *Breakout*.
- Supprimer toutes les sources du projet *Breakout* et faire une opération similaire de copie puis de comparaison mais en utilisant la vue *Package Explorer*, que remarquez-vous ?

Exercice 3 :
- Dans la vue *Navigator*, renommer les fichiers `PongFrame.java` en `BreakoutFrame.java` et `PongCanvas.java` en `BreakoutCanvas.java`.

- Comparer les fichiers correspondants de *Pong-correction* et *Breakout* (le projet fonctionne-t-il encore ?).
- Utiliser l'historique local pour restaurer `PongFrame.java` et `PongCanvas.java` dans le projet *Breakout* et supprimer les nouveaux fichiers `BreakoutFrame.java` et `BreakoutCanvas.java`.
- Cette fois-ci dans la vue *Package Explorer*, renommer (avec *Refactor / Rename...*) comme précédemment et constater les différences.
- Fermer le projet *Pong-correction*, puisqu'il est maintenant inutile pour la suite.
- Renommer la classe `Table` en `Arena` et changer également les identifiants des déclarations de variable de type `Arena` de `table` en `arena` (soit en utilisant une recherche dans tout le projet et la fonctionnalité *Word Completion* associée à la vue *Problems*, soit automatiquement...). Modifier le nom des identifiants des variables de type `PongFrame` et `PongCanvas` dans le projet si ce n'est déjà fait.
- Vérifier que votre projet *Breakout* fonctionne encore malgré tous ces changements.

Suppression du code inutile

Il s'agit maintenant de transformer réellement le projet *Breakout* en jeu de « casse-briques » : on commence par supprimer la deuxième raquette et le deuxième score (vous pouvez aussi vous contenter de commenter avec la commande *Toggle Comment*).

Exercice 4 :
- Rechercher toutes les références au champ `paddle2` de la classe `BreakoutFrame` pour les supprimer, ainsi que le code devenu inutile (gestion du l'adversaire, du vainqueur, ...).
- Changer toutes les références à `paddle1` en `paddle` (utiliser l'outil adapté d'Eclipse pour chercher ces références et la fonctionnalité *QuickFix*, ou bien alors utiliser *Refactor/ Rename...*).
- De même, rechercher toutes les références à `score2` de la classe `BreakoutFrame` pour les supprimer et enlever le code inutile.
- Changer toutes les références à `score1` en `score`.
- Finir de nettoyer en supprimant le `1` de tous les noms d'identifiants en `*[Ss]core1*` ou `*[Pp]addle1*`.
- Une fois de plus, vérifier que votre projet fonctionne encore.

Après toutes ces modifications vous ne devriez avoir qu'une seule raquette, un score et un filet.

Exercice 5 : Faire en sorte que la balle parte de la raquette (utiliser la vue *Type Hierarchy* pour localiser la méthode qui réinitialise la balle puis modifier-la pour qu'elle place la balle sur la raquette). Vous devrez rajouter un appel à cette méthode dans la méthode `initPong()`. Valider les modifications à l'aide de la vue *Problems*. et vérifier que le projet fonctionne.

Exercice 6 : Supprimer les champs inutiles d'`Arena` (ceux concernant le filet).

Exercice 7 : Changer la forme de la balle en rond : utiliser l'aperçu *Javadoc* pour comprendre comment la balle est dessinée et trouver, grâce à *Content Assist*, quelle méthode de la classe `Graphics` utiliser et comment.

Exercice 8 : En vous inspirant du code existant, rajouter la gestion des rebonds contre la paroi droite de l'arène.

Maintenant vous devriez avoir un « casse-briques », mais sans brique...

Ajout des briques

La dernière étape de la transformation du *Pong* en *Breakout* consiste à rajouter les rangées de briques, et à gérer les collisions de la balle avec ces briques et leur disparition.

Exercice 9 : Créer une nouvelle classe vide `Block` qui implémente l'interface `Drawable`. Chaque instance de cette classe représente une brique. S'inspirer de ce qui existe pour la classe `Paddle` pour le code de cette classe et l'affichage. Rajouter le code nécessaire pour l'affichage de l'ensemble des briques dans la classe principale (pour les dimensions, on pourra par exemple prendre : largeur de l'écran / 18, hauteur de l'écran / 8 et faire 8 colonnes sur 8 lignes de briques).

Exercice 10 : Donner des couleurs différentes aux colonnes de briques : rajouter le code nécessaire dans la classe `Block` et utiliser (bien sûr) *Content Assist* pour trouver les couleurs prédéfinies dans la classe `Color`.

Exercice 11 : Rajouter le code nécessaire à la gestion des rebonds de la balle sur les briques (pour l'instant uniquement les rebonds sur la face de gauche des briques) en s'inspirant du code existant pour le rebond de la balle sur la raquette. Rajouter un état `isAlive` dans la classe `Block` qui indique si la brique a été touchée ou non et en tenir compte dans l'affichage et dans la mise à jour de l'état du jeu (et ne pas oublier de mettre à jour le score à chaque rebond/destruction de brique).

Bonus

Exercice 12 : S'il reste du temps, en vrac. . . :

1. Afficher en permanence le meilleur score (le plus grand nombre de briques détruites avant de perdre).
2. Le *Breakout* a conservé l'orientation horizontale du *Pong*, faire tout pivoter pour obtenir l'orientation verticale d'un « casse-briques ».
3. Rajouter la gestion des rebonds sur les autres faces des briques.
4. Rajouter des bonus cachés dans certaines briques qui, par exemple, ralentissent momentanément la balle, ou bien augmentent la taille de la raquette, . . .