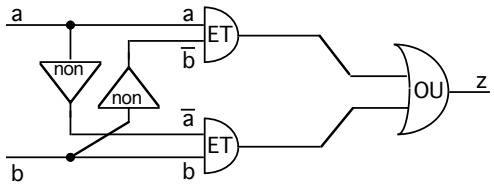
TP2 – POO (Ruby)

1) Les Circuits Logiques

On désire écrire un simulateur de circuits logiques à partir de portes élémentaires (Et, Ou, Non) à l'aide d'un langage orienté objet. Par exemple voici un circuit qui correspond à la fonction booléenne :

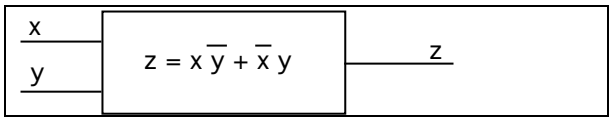
$$z = a\,\bar{b} + \bar{a}\,b$$

et qui représente un **OU-exclusif :**



a) Définissez la hierarchie de classes nécessaire à la représentation des composants. Un composant dispose d'une sortie calculée par une fonction de simulation en fonction de ses entrées. On distinguera les Composants binaires à 2 entrées des unaires à une entrée.
On illustra cette notion de composant en implémentant les classes de base **Et**, **Ou**, **Non**.
Vous pouvez définir des classes supplémentaires si vous en avez besoin. Par exemple une classe **Borne** pour représenter les valeurs booléennes en entrée et/ou sortie).

b) On veut pouvoir utiliser directement des composants et les intégrer dans des circuits plus complexes. Un circuit pouvant se définir par sa liste de composants. On pourra également opérer une classification des circuits en fonction de leur nombre de sorties.
Définissez une hierarchie de classes permettant de représenter des circuits et illustrer en implémentant la classe **Xor** (OU Exclusif) dont la fonction booléenne de transfert est, pour chacune de ses instances, à partir des composants **Et**, **Ou**, et **Non**:

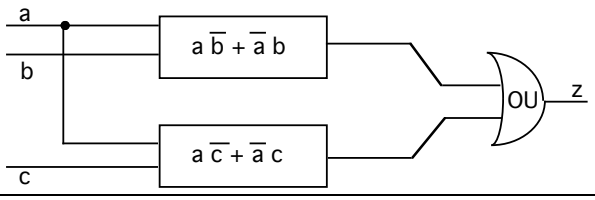


c) Créez une instance de la classe précédente, et donnez la trace de l'ensemble des envois de messages utilisés lors de la simulation, pour les valeurs suivantes des bornes : x à pour valeur True, y à pour valeur False.

d) Donnez la classe du circuit correspondant à la fonction booléenne :

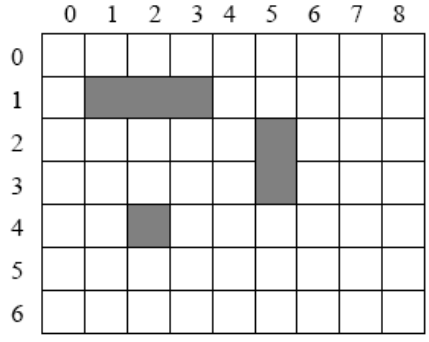
$$z = a\,\bar{b} + \bar{a}\,b + a\,\bar{c} + \bar{a}\,c$$

dont le schéma est :



2) Une Modélisation de la Bataille Navale

On veut réaliser un jeu de bataille navale. Un jeu de bataille navale se compose d'une grille et d'un ensemble de bateaux, chaque bateau se compose d'un ensemble de taille fixe d'éléments. Un **escorteur** comprend 3 éléments, un **remorqueur** 2 et un **sous-marin** un seul élément.
Chaque élément est caractérisé par sa position et par son état : sain ou touché.
Une grille contient un ensemble de bateaux (stockés dans une Liste). Un bateau est caractérisé par l'ensemble de ses éléments (stockés dans un tableau).

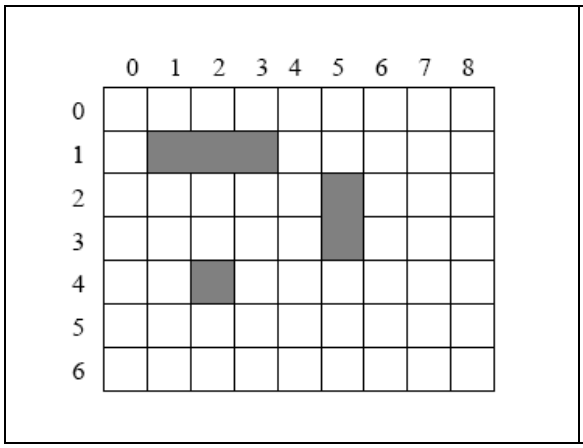


Les **sous-marins** ont la possibilité de plonger. Lorsqu'ils plongent ils ne peuvent pas être touchés.
Les **escorteurs** disposent de canons tandis que les **remorqueurs** sont caractérisés par leur vitesse d'intervention (un

coefficient multiplicateur / aux autres bateaux et leur puissance de remorquage
Voici comment on instancie une flotte de bateaux (qui correspond à la figure) :

```
# un escorteur horizontal dont le premier élément est en 1,1 et qui dispose
# de 20 canons.
# (les coordonnées ont leur origine en 0,0).
bateau1 = Escorteur.placer(1,1, true, 20);
# un Remorqueur vertical dont le premier élément est en 2,5 deux fois plus
# rapide et de puissance 10000 Carambars (unités de puissance des
# remorqueurs)
bateau2 = Remorqueur.placer(2,5,false, 2, 10000);
# un sous-marin en 4,2 et en surface (false = en plongée).
bateau3 = SousMarin.placer(4,2,true);
# le plateau du jeu
grille1 = Grille.construire(7,9);
grille1.ajouterBateau(bateau1);
grille1.ajouterBateau(bateau2);
grille1.ajouterBateau(bateau3);
```

1. Donnez la liste des classes nécessaires et leurs caractéristiques principales (vi et méthodes)
2. Donnez le code des méthodes `ajouterBateau` et `enleverBateau` dans `Grille`.
3. Donnez le code de la méthode `estTouche?(px, py)` définie dans Bateau (et éventuellement dans ses sous-classes) qui retourne 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément touché, 2 si le coup tombe sur un élément touché pour la première fois, et 3 si le bateau est coulé (tous les éléments sont touchés).
4. Donnez en Ruby le code complet des classes Bateau définies (variables d'instances, création / initialisation, méthodes)
5. Donnez le code de la méthode `coup(px, py)` définie dans `Grille`, qui retourne -1 si le coup est en dehors de la grille, 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément déjà touché, 2 si le coup tombe sur un élément touché pour la première fois et 3 si le bateau est coulé. Si le bateau est coulé, il est alors supprimé de la grille.
6. On suppose que les bateaux peuvent avancer, sauf lorsque les sous-marins sont en plongée. Ecrire la méthode `avancer?(dx, dy)` dans `Bateau` (et éventuellement dans ses sous-classes) qui fait avancer le bateau dans la direction définie par `dx` et `dy` (la valeur de `dx` et `dy` est -1 ou 1, ce qui implique que les bateaux peuvent avancer en diagonale). Cette méthode renvoie un booléen qui indique si le déplacement est possible.
7. Les bateaux sont comparable deux a deux le critère de comparaison étant le pourcentage d'élément du bateau en bon état.
8. Ecrire le code Ruby Nécessaire à l'affichage de la grille (sous forme textuelle) en utilisant des * pour séparer les Lignes et les Colonnes, E pour un élément d'un escorteur, S pour un Sous-Marin et R pour un remorqueur.




```
   0   1   2   3   4   5   6   7   8   9
 *****************************************
0*   *   *   *   *   *   *   *   *   *   *
 *****************************************
1*   * E * E * E *   *   *   *   *   *   *
 *****************************************
2*   *   *   *   *   * R *   *   *   *   *
 *****************************************
3*   *   *   *   *   * R *   *   *   *   *
 *****************************************
4*   *   * S *   *   *   *   *   *   *   *
 *****************************************
5*   *   *   *   *   *   *   *   *   *   *
 *****************************************
6*   *   *   *   *   *   *   *   *   *   *
 *****************************************
```