Exercice1 : Il s'agit de modéliser un segment de droite dont les valeurs des deux extrémités sont entières. Si on échange les deux extrémités, on considère qu'il s'agit encore du même segment. La classe Segment comporte les méthodes suivantes :

- un constructeur de ce segment recevant en argument les deux valeurs entières des extrémités du segment que l'on veut construire .
- une méthode retournant la longueur du segment.
- une méthode nommée ordonne échangeant éventuellement les valeurs des extrémités du segment de telle sorte que la valeur de extr1 soit au plus égale à la valeur de extr2 .
- une méthode testant si un entier donné se trouve sur le segment (c'est-à-dire s'il est compris entre la plus petite et la plus grande valeur des extrémités du segment).
- public String toString() :Celle-ci décrira une instance de Segment sous la forme d'une chaîne de caractères (par exemple, le segment d'extrémités -35 et 44 pourra être transformé en la chaîne de caractères : "segment [-35, 44]") (la plus petite extrémité est toujours indiquée à gauche).

-Ecrivez aussi une autre classe EssaiSegment comportant la méthode main et permettant de tester les méthodes de la classe Segment .

Exercice2 : Ecrivez une classe Complexe représentant des nombres complexes. Un nombre complexe comporte une partie réelle et une partie imaginaire (partieReelle + partieImaginaire * i .

-La classe Complexe doit disposer des constructeurs suivants: Complexe(): constructeur par défaut, Complexe(partieReelle, partieImaginaire),Complexe(Complexe).

-Elle doit contenir des accesseurs et mutateurs pour les différents attributs.

-additionner au nombre complexe concerné un autre complexe ; on écrira pour cela une méthode d'instance nommée addition qui recevra en paramètre l'autre complexe et qui ne retournera rien.

- additionner deux complexes donnés ; on écrira pour cela une méthode statique nommée aussi addition (en utilisant ainsi la possibilité de la surcharge) qui recevra en paramètres les deux nombres complexes à additionner et qui retournera le résultat sous forme d'un objet de type Complexe.

-une méthode toString() donnant une représentation d'un nombre complexe (a+b*i).

-Ecrivez aussi une classe testComplexe afin de tester la classe Complexe.

Exercice3: Ecrivez un programme Geometrie qui permet à l'utilisateur d'entrer les coordonnées (x, y) des sommets d'un triangle. Le programme affiche ensuite le périmètre du triangle ainsi qu'un message indiquant s'il s'agit d'un triangle isocèle. Votre programme doit être orienté objet. Indications:

- Un triangle est isocèle si au moins deux côtés ont la même longueur.
- La formule pour calculer la distance entre deux points (x1, y1) et (x2, y2) est: racine carrée de $(x1 - x2)^2 + (y1 - y2)^2$.

Exercice4: On veut gérer les guichets d'une banque. Les guichets sont numérotés de 0 à N -1.Quand un client arrive, on doit lui indiquer quel guichet est libre. On représente les guichets par un tableau de booléens qui indique que le guichet i est libre (true) ou non (false).

1- Définir une classe GuichetsBanque qui a un champ tabGuichets et un constructeur qui prend comme paramètre un nombre de guichets N et initialise tabGuichets avec une taille N et true partout.

2- Rajouter une méthode guichetLibre qui renvoie le numéro du premier guichet libre. Si aucun guichet n'est libre, la fonction affiche un message d'erreur et renvoie -1.

3- Rajouter Une fonction qui permet de modifier l'état d'un guichet : void modifEtat(int num).

4- Rajouter une fonction qui permet de déterminer le nombre total de guichets libres : int nbGuichetLibre( ) ;

Exercice5: Problème (Jeu des petits chevaux)

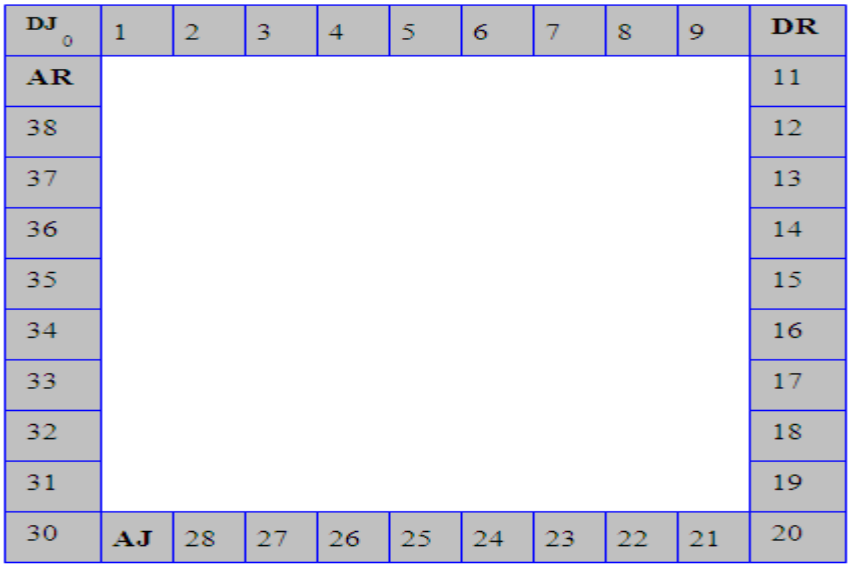
On pose :

DJ = Depart jaune

DR= Depart rouge

AJ = Arrivée jaune

AR = Arrivée rouge



Les règles du jeu des petits chevaux auquel, on joue ici sont les suivantes :

- Le jeu se joue à deux joueurs, chaque joueur a une couleur différente (rouge ou jaune)
- Chaque joueur possède 2 pions, 1 lent et 1 rapide. Lorsque c'est son tour de jouer, le joueur choisit un pion, et, lance un dé si le pion est lent et 2 dès si le pion est rapide. Le pion avance ensuite de la valeur du ou des dès.

- Le plateau de jeu est comme sur la figure . C'est une grille de taille 40 (voir la figure), qui comporte 2 points de départ (DJ et DR) et deux points d'arrivée (AJ et AR).
- Un pion est arrivé lorsque sa position est supérieure à la position de l'arrivée.
- Un joueur a gagné lorsque ses deux pions sont arrivés.

A- Les pions

On va d'abord définir une classe Pion dont le squelette est donné par :

```
class Pion{
        String couleur;
        boolean vitesse ;
       int position ;
// …… autres codes de la classe
void avancer (int n){//…}
static void avancer(int n, Pion p){//…}
void avancerAvecConflit(int n, Pion p){//…}
boolean estArrive(){//…}
}
```

1- Ecrire un premier constructeur pour la classe Pion qui prend en argument la couleur, la vitesse et la position de l'objet qu'on veut créer.

2- Ecrire un deuxième constructeur qui prend en argument la couleur et la vitesse, et initialise la position correctement

3- Ecrire la méthode estArrive qui renvoi true si le pion est arrivé, et false sinon.

4- Ecrire la méthode avancer(int n) doit faire avancer le pion de n cases, sans le faire dépasser la ligne d'arrivée.

5- Ecrire la version static de cette méthode.

6- Ecrire la méthode void avancerAvecConflit(int n, Pion pio), qui fait la même chose que la méthodes avancer de la classe Pion et qui prend compte de cette modification Si un pion arrive sur une case où se trouve un pion adverse, il faut le renvoyer à sa case de départ.

B- La classe Jeu va utiliser la classe Pion. Il est donc vivement recommandé d'utiliser les méthodes de la classe Pion.

Le squelette de la classe Jeu est donné par :

```
class Jeu{
Pion jauneLent ;
        Pion jauneRapide ;
        Pion rougeLent ;
        Pion rougerapide ;
```

```
        // …… autres codes de la classe
        void jouer(String couleur){}
        void jouerModifier(String couleur){}
boolean aGagne(String couleur){}
}
```

Pour l'instant les deux joueurs cohabitent paisiblement sur le plateau de jeu.

1- Ecrire un constructeur pour la classe Jeu. Sachant que la vitesse d'un pion lent est égale à false ;

2- Ecrire la méthode jouer, qui décide aléatoirement quel pion avance et tire au hasard le nombre de case dont le pion avance. On rappelle que les pions ont des vitesses différentes

3- Ecrire la méthode jouerModifier, qui fait la même chose que la méthode jouer mais comprend en compte le fait de ne pas faire jouer un pion qui est déjà arrivé

4- Ecrire la méthode aGagne. Un joueur a gagné si ses deux pions sont arrivés

5- Ecrire une méthode qui fait jouer les joueurs alternativement jusqu'à la victoire d'un des deux joueurs : void jeuComplet() ;