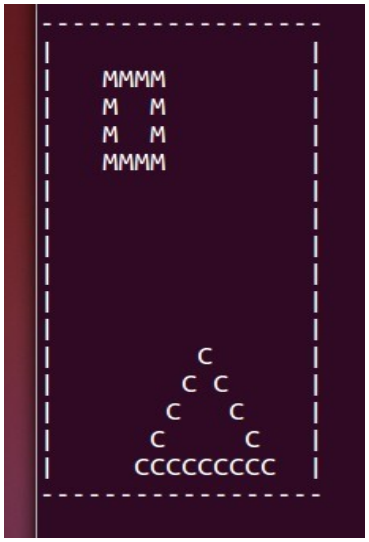
Un ensemble de points colorés représentés comme des caractères dans une matrice représentant un écran



Exposé du problème

Le thème est celui d'une application qui simule un affichage de points dans un écran :

- les points sont des points de couleur dont l'affichage consiste à afficher le caractère initial de leur couleur
  - on se limite à 4 couleurs : le cyan, le magenta, le jaune et le noir (système CMJN)
- l'écran est une matrice de caractères (tableau à 2 indices : ligne et colonne)

L'espace des réalisations possibles étant sans limites, l'exercice propose de compléter une application à partir de certains éléments déjà fixés.

À cet effet, différents fichiers de l'application sont téléchargeables :

- un exécutable sous la forme des 4 fichiers ".class" que doit comporter l'application
  - TestEcran.class (fichier principal)
  - IhmAvecMenu.class
  - Ecran.class
  - Point.class
- les fichiers source
  - TestEcran.java
  - IhmAvecMenu.java

Par ailleurs, on fixe la partie structure de données de la classe Ecran à :

```
// ****************************
// *****  structure de données *****
// ****************************

// dimensions de l'Ecran
private int nbLignes ;
private int nbColonnes ;

// liste des Point de l'Ecran de l'application
private ArrayList<Point> listeDesPoints ;
```

Un aperçu de la classe *ArrayList* fournie par Java est donné en annexe (un aperçu réduit aux besoins de l'exercice).

L'exercice proposé

Il est demandé de compléter les sources de l'application de sorte que l'exécution de celle-ci soit conforme dans ses résultats à l'exécutable livré.

On veillera à respecter le principe d'encapsulation :

- tous les éléments de la structure de données d'une classe sont déclarés privés
- on évite autant que possible, voire totalement, la définition d'accesseurs en lecture (les *getters* des environnements de programmation Java)

Annexe : quelques mots sur la classe ArrayList

- la classe *ArrayList* de Java est une ***classe générique***, c'est-à-dire qu'elle est présentée comme *Arraylist<E>* où *E* est un paramètre formel de classe :
  - il faut importer la classe *ArrayList*, ce qu'on fait grâce à la commande
    import java.util.ArrayList ;
    placée en tête de fichier avant la définition de la classe qui l'utilise
  - pour définir une variable objet de la classe *ArrayList*, on doit fournir un paramètre effectif pour *E*, comme par exemple dans la classe *Ecran* de l'application proposée :
    private ArrayList<Point> listeDesPoints ;
  - le paramètre effectif fourni doit être un nom de classe :

ArrayList<Point>	OK, *Point* est une classe de l'application
ArrayList<int>	pas OK, *int* est un type de base, pas une classe
ArrayList<Integer>	OK, *Integer* est la classe Java implémentant les entiers

- on peut voir les *ArrayList* tout aussi bien :
  - comme des tableaux (structure de données à accès direct, les éléments sont indicés par par l'intervalle d'entiers [0 ; this.length-1]
  - comme des listes (structures à accès séquentiel, muni de possibilités d'insertion et de suppression)
  - mais dont tous les éléments doivent être de même type au sens qu'ils doivent référencer des objets héritant tous de la classe fournie comme paramètre effectif pour la déclaration de l'*ArrayList*

On peut consulter le site officiel
http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html
pour une information complète, mais voici quelques méthodes suffisant a priori à assurer la réalisation de l'application demandée :

- un constructeur sans paramètres *ArrayList*()
  - retourne une référence sur un *ArrayList* vide
  - exemple
    - après la déclaration de l'attribut de la classe *Ecran*
      private ArrayList<Point> listeDesPoints ;
    - à la création d'un objet instance de la classe *Ecran*, la création de son attribut
      this.listeDesPoints = new ArrayList<Point>() ;

- une méthode pour ajouter un élément à un *ArrayList<E>*
  - public void add(E element)
    - ajoute l'objet référencé par la variable objet *element* de la classe *E* comme dernier élément de l'*ArrayList*

- une méthode pour supprimer un élément à un *ArrayList*
  - public void remove(E element)
    - supprime de l'*ArrayList* la première occurrence, si elle existe, de l'objet référencé par la variable objet *element* de la classe *E*

- une méthode pour vider un *ArrayList*
  - public void clear()
    - supprime tous les éléments de l'*ArrayList*

- une méthode pour parcourir un *ArrayList*
  - avec l'exemple de la liste des *Point* d'un *Ecran*
    - for (Point p : this.listeDesPoints)
      {
      */ dans tout ce bloc, la variable objet *p*, déclarée de la classe *Point*, prend successivement les valeurs de tous les *Point* de l'*ArrayList* this.listeDesPoints */
      …
      }