**JAVA TD**

**( ? )** Question : Pourquoi le vecteur ne déclare pas un type déterminé ? 🡪 pour utiliser la classe abstraite FormeColoree

On a Vector<int>, Vector<Shape>…pourtant dans la classe VisualiseurDeFormes les codes montrent Vector m\_formes, sans type précisé.

**Exercice 1 : Classe, Hiérachie**

1. **Repérez des classes, indiquez si elle hérite d’une autre :**  
   - la classe Cercle  
   - la classe VisualiseurDeFormes hérite de la classe JFrame  
   - la classe ZoneDeDessin hérite de la classe JPanel
2. **Pourquoi Forme est-elle une interface et non une classe :**  
   - l’entête de la fonction redessiner( Graphics g ) peut être utilisée par plusieurs classes 🡪 cette fonction sera surdéfinie par d’autres classes de façon différente.  
   - Forme est presque une classe abstraite : il s’agit d’une conception abstraite qui n’a pas de précis attributs ( une Forme peut être un Cercle, un Rectangle, un Triangle… ) et dont les subclasses se comportent différemment.
3. **Quelle est la classe qui représente la fenêtre de l’application ? Changez donc le titre affiché sur la fenêtre :**- c’est la classe VisualiseurDeFormes, car :   
    + le programme principal ( public static void main ) s’écrit dans le corps de cette classe  
    + dans son constructeur, il y a des instructions qui servent à définie une application ( déterminer la taille, gérer des événements de boutons cliqués, de la fin de l’application… )  
     
   - pour changer le titre affiché sur la fenêtre ( « un très joli visualiseur » ) : on modifie l’instruction super(« … » ) dans le constructeur.
4. **La classe qui se charge de dessiner les cercles ?** 🡪 la classe ZoneDeDessin  
   **La méthode òu l’affichage des formes/cercles est réalisée** 🡪 public void paintComponent( Graphics g )

**Exercice 2 : Objets**

Tous les objets instanciés dans le constructeur de VisualiseurDeFormes  et leur utilité :

* m\_formes = new **Vector**( ) 🡪 un conteneur sert à stocker les formes
* m\_random = new **Random**( ) 🡪 créer un nouveau générateur de nombres aléatoires
* new **ZoneDeDessin**( ) 🡪 instancier la classe qui charge de dessiner les formes
* m\_panneau\_boutons = new **JPanel**( ) 🡪 creates a new JPanel with double buffers & flow layout
* label = new **JLabel**( "Visualiseur de Formes" ) 🡪 créer une instance de la classe JLabel avec le texte spécifié entre les « … »
* new **BoutonCercleAction**( this ) 🡪créer une instance de la classe BoutonCercleAction qui modélise la création d’un cercle suite à un clic sur le bouton
* new **JButton**( "Cercle" ) 🡪 créer un nouveau bouton avec le texte spécifié entre les «… »

Quelles sont les relations entre ces objets ?

* Random ( m\_random )– BoutonCercleAction ( action\_cercle ) : dépendance
* BoutonCercleAction ( action\_cercle ) – Jbutton ( b1 ) : action\_cercle gère l’événement quand l’utilisateur appuie sur le bouton b1 de nom « Cercle »
* ZoneDeDessin ( m\_panneau\_dessin ) – Vector ( m\_formes ) : l’objet m\_panneau\_dessin réfère au vecteur stockant les formes pour les dessiner

**Exercice 3 : Programme principal**

* La méthode main se trouve dans la classe VisualiseurDeFormes. Pour créer une fêntre graphique, on a besoin une classe qui hérite de la classe JFrame.  
  Elle crée la fenêtre graphique en faisant l’allocation VisualiseurDeFormes visu = new VisualiseurDeFormes( ), puis on l’affiche en utilisant la fonction setVisible( true ) ;
* Faire 2 instances de VisualiseurDeFormes pour créer 2 fenêtres d’application.
* Pour que le titre des fenêtres soit différent :
  + on ajoute le paramètre String title dans le constructeur de la classe VisualiseurDeFormes
  + la variable statique

**Exercice 4 : Polymorphisme : d’autres formes**

* **Les cercles et rectangles sont stockés dans un même objet (un vecteur d’Objet) et pourtant leur affichage est bien différencié. Comment cela est-il possible ?**  
  🡪 Parce que ces classes héritent d’une même interface Forme. Il s’agit bien d’un Vecteur contenant les pointeurs de type Forme, et chaque pointeur prend l’adresse d’un objet plus précis des sous-classes.
* **Que se passe-t-il si vous enlevez dans la méthode ZoneDeDessin.paintComponent le terme (Forme) devant m\_formes.get( i ) ?**  
  🡪 Le compilateur signale des erreurs : *Type mismatch ( cannot convert from Object to Forme )*🡪 Il faut un transtypage pour transformer l’objet donné par m\_formes.get( i ) au type plus précis ( Forme )
* **m\_formes.add( new String( "Texte" ) );  
  Le programme JAVA compile-t-il ? Si oui, que se passe-t-il à l’exécution ?**  
  🡪 Oui, il compile car en ce moment il s’agit d’un Vecteur de l’Objet, String est ainsi un Objet, pourtant, une exception est lancée :   
  *Exception in thread "AWT-EventQueue-0" java.lang.ClassCastException: java.lang.****String cannot be cast to Forme at ZoneDeDessin.paintComponent****(ZoneDeDessin.java:36)*

**Exercice 5 : Bouton de création de rectangles**

**Exercice 6 : Bouton « reset »**

**Exercice 7 : Bouton « aléatoire »**

**Exercice 8 : Forme colorée**

* Peut-on rajouter une donnée membre à l’interface Forme ?  
  🡪 Non, parce qu’on ne peut pas déclarer un attribut dans l’interface Forme.
* Si non, que pensez-vous de la technique consistant à rajouter une donnée membre couleur à chaque sous-classe de Forme ?  
  🡪 Au lieu d’une interface Forme, on modifie la hiérachie d’héritage de manière que la super-classe maintenant soit la classe abstraite FormeColoree, et toutes les sous-classes ( Cercle, Rectangle… ) héritent de FormeColoree.
* Pourquoi la classe FormeColoree doit-elle être abstraite ?  
  🡪 Il s’agit d’une conception abstraite, une forme générale n’a aucun attribut précis ( un rayon ? une longueur et une largeur ? …). De plus, la fonction redessiner( Graphics g ) sera utilisée par plusieurs sous-classes, et elle sera redéfinie à chaque fois. 🡪 le polymorphisme

**Exercice 9 : Classes imbriquées**

**Exercice 10 : IHM, création de classe dérivée : rectangle de position et taille spécifiées à la souris**

WHAT