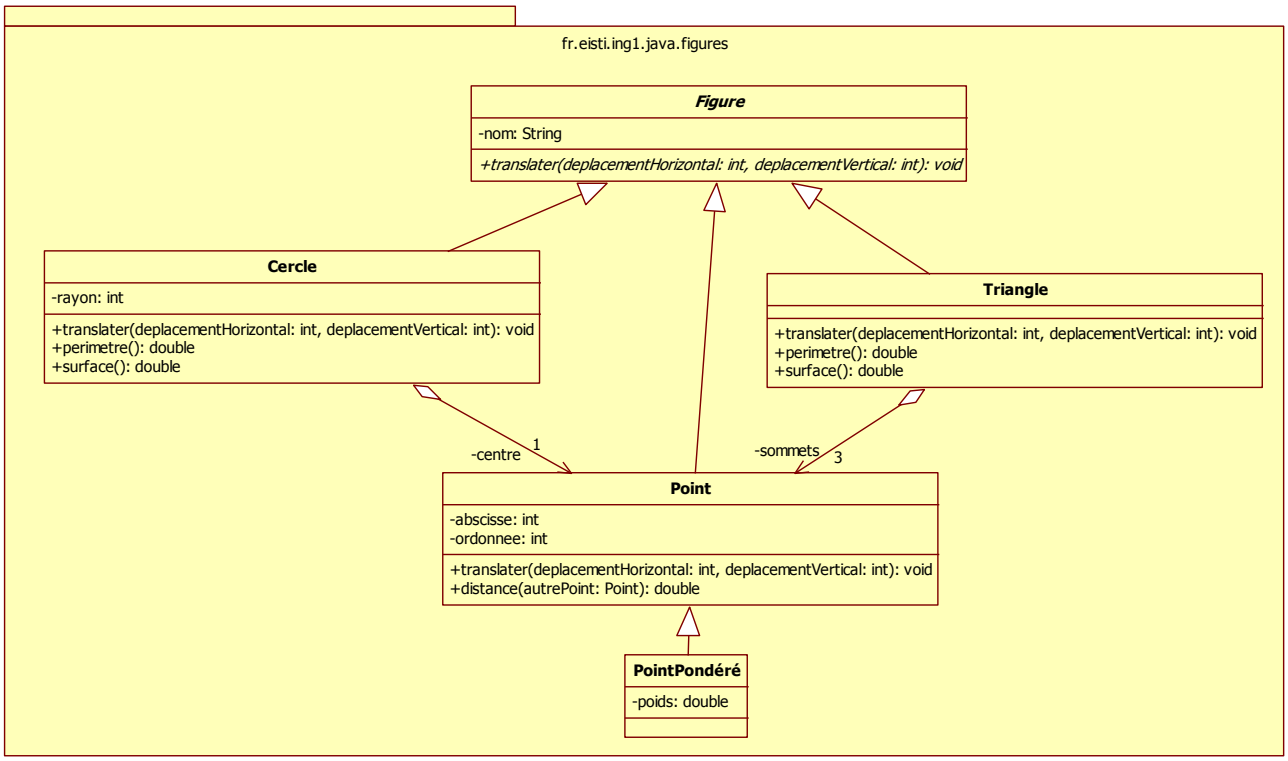
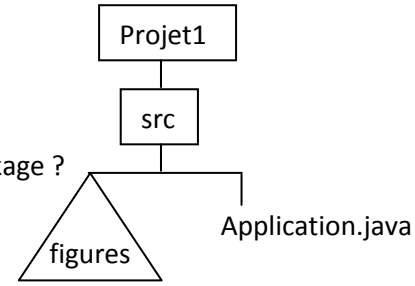
Introduction.

Le but de ce TD est de produire une librairie sous la forme d'un package contenant toutes les figures écrites dans les séances précédentes.



Exercice 1. Package fr.eisti.ing1.java.figures



1.1 – Créer le package fr.eisti.ing1.java.figures regroupant les classes Figure, Triangle, Point, Cercle et PointPondéré :

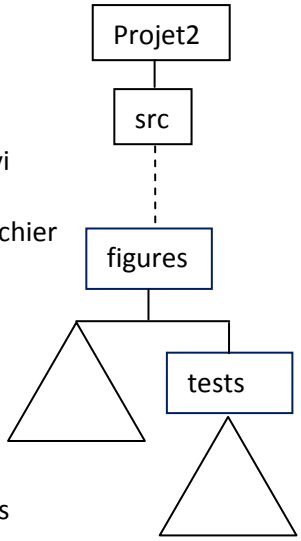
- **1.1.1** Quel est le nom du répertoire qui contient les classes du package ?
- **1.1.2** Modifier les différentes classes pour tenir compte de l'appartenance au package

1.2 – Comment compiler en ligne de commande les classes du package depuis le répertoire du projet ? En déduire les modifications éventuelles de votre fichier ant.

1.3 – Laisser votre application manipulant vos figures à la racine de vos sources.

- **1.3.1** Comment utiliser les classes du package dans votre application ? Faites les modifications nécessaires.
- **1.3.2** Comment compiler à la fois l'application et le package ? Tester en ligne de commande puis modifier éventuellement votre fichier ant.

Exercice 2. Exporter un package



Pour pouvoir distribuer un package, quelques critères sont à prendre en compte :

- le package doit avoir un nom unique : en général, un nom de domaine suivi [d'un nom de département/projet puis] du nom du package
- on préfèrera une version archivée en jar pour pouvoir manipuler un seul fichier plutôt que *n* fichiers .class + 1 arborescence

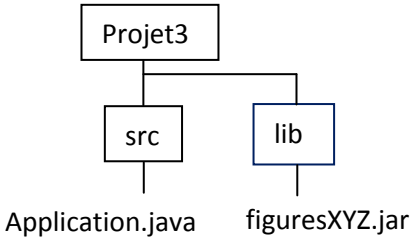
2.1 – Choisir un nom de package unique qui vous démarque de vos camarades. Créer un projet pour gérer uniquement ce package. Placer vos différents tests dans un sous-package .....figures.tests.

- **2.1.1** – Ecrire des cibles ant qui permettent de compiler les figures et vos tests et de les exécuter.
- **2.1.2** – Ecrire des cibles ant qui permettent de compiler uniquement les classes du packages puis archive le bytecode dans un fichier .jar.

2.2 – Pour tester la version archivée de votre package :

2.2.1 – Créer un nouveau projet avec une seule classe application

2.2.2 – Placer le fichier jar contenant votre package dans un répertoire lib à la racine de votre projet :



2.2.3 – Compiler et exécuter votre application en ligne de commande

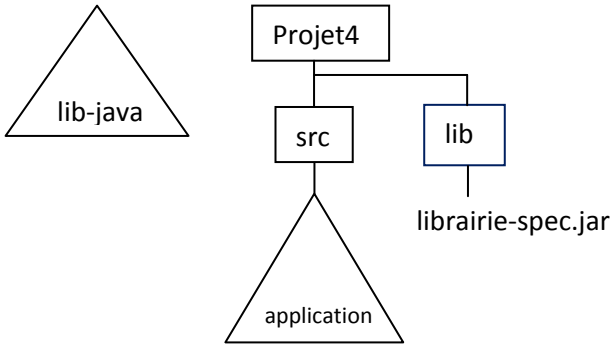
2.2.4 – Ecrire un fichier ant qui vous permette d'automatiser ces 2 tâches

Exercice 3. Une application de très grande taille

Pour développer une application objet en Java, on réutilise des classes existantes en grand nombre et de provenances multiples : API Java, Oracle, Apache, etc, ... La plupart des packages utilisés sont disponibles sous la forme d'archives jar. En général, on place ces archives dans une arborescence dédiée si on les utilise dans de nombreux projets ou dans un sous-répertoire du projet pour un usage unique.

Récupérer sur Arel l'archive **fichiers-td6.tgz** contenant :

- les sources d'une application séparée en plusieurs packages : fr.eisti.application, fr.eisti.application.module1 et fr.eisti.application.module2.
- une librairie spécifique à ce projet lib/librairie-spec.jar.
- un répertoire contenant un grand nombre de librairies que l'on considère utile au développement de plusieurs projets : lib-java. Déplacer ce répertoire dans une destination qui vous semble adéquate (la racine de votre compte par exemple).



Ecrire un fichier ant qui permet de :

3.1. Compiler le projet en prenant en compte toutes les librairies utilisées dans les sources. Préférer les chemins absolus pour référencer les librairies générales. Dans un shell, le classpath devient difficile à manipuler !

Heureusement, ant vous permettra de le construire facilement.

```
mkdir -p build/classes
javac -d build/classes -source-path src \
-classpath lib/librairie-spec.jar:.../lib-java/impression.jar:.../lib-java/mail
.jar:.../lib-java/ojdbc14.jar:.../lib-java/webservices/wsdl.jar:.../lib-java/s
oap.jar:.../lib-java /webservices/xmlparser.jar:.../lib-java /http.jar:.../lib
-java /outils/boite-outils1.jar:.../lib-java /outils/boite-outils2.jar:.../lib-
java /outils/boite-outils3.jar:.../lib-java /outils/boite-outils4.jar:.../lib-j
ava/outils/boite-outils5.jar:.../lib-java/outils/boite-outils6.jar:.../lib-java
/outils/boite-outils7.jar:.../lib-java/outils/boite-outils8.jar:.../lib-java/ou
tils/boite-outils9.jar:.../lib-java/outils/boite-outils10.jar \
src/fr/eisti/application/Application.java
```

3.2. Exécuter le projet :

```
java -cp build/classes:lib/librairie-spec.jar:.../lib-java/impression.jar:.../l
ib-java/mail.jar:.../lib-java/ojdbc14.jar:.../lib-java/webservices/wsdl.jar:...
./lib-java/soap.jar:.../lib-java /webservices/xmlparser.jar:.../lib-java /http.
jar:.../lib-java /outils/boite-outils1.jar:.../lib-java /outils/boite-outils2.j
ar:.../lib-java /outils/boite-outils3.jar:.../lib-java /outils/boite-outils4.ja
r:.../lib-java/outils/boite-outils5.jar:.../lib-java/outils/boite-outils6.jar:.
../lib-java/outils/boite-outils7.jar:.../lib-java/outils/boite-outils8.jar:...
/lib-java/outils/boite-outils9.jar:.../lib-java/outils/boite-outils10.jar \
fr.eisti.application.Application
```