(M2103) Test 1 de POO

Le 4 mai 2019– durée : 2h

© *Marc Laporte marc.laporte@univ-amu.fr*
Aix Marseille Université
I.U.T. d'Aix en Provence - Département Informatique

Preambule

Documents autorisés : une feuille simple A4 manuscrite recto / verso.
Remarques :

- Lire attentivement tout le sujet, avant de commencer. Le barème, donné à titre indicatif, est sur 21 points, et 0 de bonus ;
- Écriture au crayon de papier interdite.

Exercice 1.

Pour schématiser, un nombre complexe est un nombre dont le carré peut être négatif.
Un nombre complexe `z` peut s'écrire de 2 façons différentes :

- en forme algébrique `z = a + i b` où `a` est la partie réelle et `b` est la partie imaginaire ;
- en forme polaire `z = ro (cos (theta) + i sin (theta))`, `ro` est appelé le module et `theta` est l'argument.

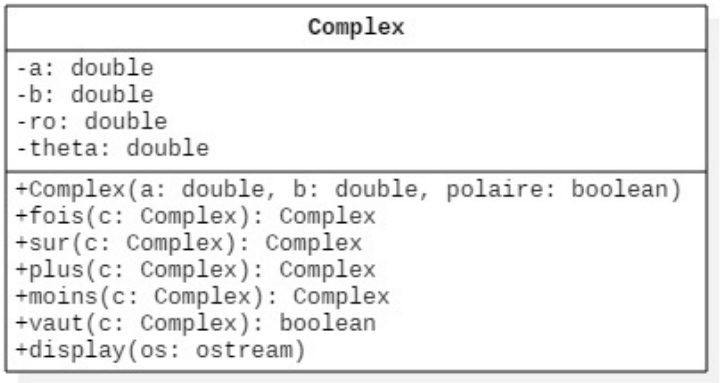
Pour passer de la forme algébrique à la forme polaire il suffit d'appliquer les calculs suivants :

- `ro = sqrt (a * a + b * b)` (`sqrt()` est la fonction racine carrée) ;
- `theta = asin (b / ro)` si `a >= 0` ou `pi - asin (b / ro)` si `a <0` (`asin()` est la fonction `arcsinus`.

Inversement pour passer de la forme polaire à la forme algébrique, il suffit d'appliquer les calculs suivant :

- `a = ro * cos (theta)`
- `b = ro * sin (theta)`

`pi` et les fonctions `sin`, `cos`, `sqrt` et `asin` sont définies dans la bibliothèque `cmath`. Le but de ce test est d'écrire une class `Complex` simplifiée (on ne signalera pas certains cas exceptionnels) dont le diagramme de classe est le suivant :



Les méthodes `fois`, `sur`, `plus`, `moins` et `vaut` seront respectivement transformées en les opérateurs `*`, `/`, `+`, `-` et `==`.
Aucune fonction n'est susceptible de lever d'exception sauf la division (fonction `sur()` dans le diagramme de classe) qui peut lever une exception du type `runtime_error`, dont le seul constructeur a comme paramètre une *NTCTS*, et la fonction `display()` qui peut lever n'importe quelle exception puisqu'elle manipule un flux.
Le constucteur doit pouvoir servir de constructeur par défaut, et dans ce cas les deux premiers paramètres (par défaut `0.0`) seront les valeurs de la forme algébrique.

Question 1.1 (7 points) :

Ecrire complètement le fichier `Complex.h` qui contient les déclarations de la classe `Complex` correspondant au schéma et aux explications (ne pas oublier de transformer les méthodes indiquées ci-dessus en *opérateurs*).

Question 1.1 (7 points) :

A l'aide des indications suivantes, écrire le fichier complet `Complex.cpp` :

- si le troisième paramètre du constructeur est `true`, les deux premiers seront les valeurs de `ro` et `theta` (forme polaires), `a` et `b` devront être calculés avec les indications précédentes ; dans le cas contraires les 2 premiers paramètres seront les valeurs de `a` et `b` (forme algébrique), et ce sont `ro` et `theta` qui devront être calculés ;
- dans les méthodes qui suivent, tous les calculs et les affichages se feront sur les attributs de la frome algébrique (`a` et `b`), `c1` sera `a1 + i * b1` et `c2` sera `a2 + i * b2` ;
- `c = c1 * c2` => `c = (a1 * a2 - b1 * b2) + i * (b1 * a2 + b2 * a1)` ;
- `c = c1 + c2` => `c = (a1 + a2) + i * (b1 + b2)` ;
- `c = c1 - c2` => `c = (a1 - a2) + i * (b1 - b2)` ;
- `c1 == c2` si `a1 == a2` et `b1 == b2` ;
- si `a1` vaut `1.0` et `b1` vaut `2.0` `c1.display ()` affichera `1.0 + i * 2.0`, `os` est le flux dans lequel on injectera ces affichages ;
- pour la division (`c = c1 / c2`) :
  - on calcule le `denominateur` qui vaut `a2 * a2 - b2 * b2 + 2 * a2 * b2` ;
  - si ce `denominateur` est nul on lève l'excetpion prévue dans le profil ;
  - `c = ((a1 * a2 - b1 * b2) / denominateur) + i * ((a1 * b2 + a2 * b1) / denominateur)`.

Question 1.1 (7 points) :

Pour tester cette classe `Complex`, écrire un morceau de programme qui : :

- déclare un `Complex (1.0, 2.0)`, en forme algébrique et l'affiche ;
- déclare un `Complex (3.0, pi)`, en forme polaire et l'affiche ;
- déclare un `Complex` nul (constructeur par défaut) et l'affiche ;
- affiche le résultat du produit des deux premiers `Complex` déclarés ci-dessus ;
- essaie d'afficher le résultat de la divission des deux derniers `Complex` déclarés ci-dessus ;
- si la division s'est mal passée affiche l'erreur ;
- affiche le résultat du test de l'égalité des 2 derniers `Complex` déclarés ci-dessus.