**Héritage, masquage de méthode**

**Exercice n°3 – Quelles méthodes hérite-t-on ? Que faut-il redéfinir ?**

**Objectif :**  
Identifier quelles méthodes de Rectangle sont héritées dans SlantedRectangle, et **les redéfinir si nécessaire**.

**Explication :**  
Par défaut, la classe SlantedRectangle hérite de **toutes les méthodes publiques** et protected de la classe Rectangle :

* **surface()**
* **translate()**
* **contains(...)**
* **equals(...)**
* **etc.**

**Mais : certaines méthodes ne sont plus correctes avec un rectangle incliné.**  
Par exemple :

* contains(Point p) : le test d’appartenance ne marche plus en supposant que les côtés sont parallèles aux axes.
* equals() : Ce ne sera plus pareil car il faut tenir compte de l’inclinaison.

Donc :

* On va **redéfinir (override)** les méthodes qui ne fonctionnent plus correctement dans un rectangle incliné.
* On doit **ajouter la rotation dans toString()** pour pouvoir visualiser l’angle

Pour le deuxième volet de la question, qui consiste à redéfinir les méthodes qui ne sont plus correctes, nous les avons implémentées dans la classe **SlantedRectangle***.*

**Exercice 4**

**Enoncer**

Pour chacun des appels de méthode ci-dessous, disons s'il va être compilé correctement et auquel cas, quelle méthode est appelée effectivement à l'exécution ?

**Solution**

* + **Point p = new Point (1, 2) ;**

**Compile normalement. On crée un Point en faisant appel au constructeur Point prenant en paramètre deux valeurs.**

* + **Rectangle r = new Rectangle (p, 2, 3) :**

**Compile correctement. On crée un rectangle et en faisant appel au constructeur Rectangle qui prend en paramètre un Point, une hauteur et une largeur ;**

* + **Rectangle t = new SlantedRectangle (p, 2, 3) ;**

**Ne compile pas.**

* + **SlantededRectangle s = new SlantedRectangle (p, 2, 3) ;**

**Ne compile pas.**

* + **System.out.println(r.surface()) ;**

**Compile correctement. On fait appel à la méthode surface() ;**

* + **r.rotate(2) ;**

**Ne compile pas car la méthode rotate() n’existe pas dans rectangle**

* + **System.out.println(r.contains(p))**

**Compile correctement. On fait appel à la méthode Contains(). Pour vérifier qu’un rectangle contient un point ;**

* + **System.out.println(t.surface()) ;**

**Compile normalement. On fait appel a la méthode surface() pour calculer la surface du rectangle t. Mais comme le rectangle t avait mal été créé cette ligne ne donnera pas un résultat en moins qu’on corrige au niveau de la création de notre Rectangle t ;**

* + **t.rotate(2) ;**

**Ne compile pas. Car la méthode rotate() n’existe pas dans Rectangle ;**

* + **System.out.println(t.contains(p)) ;**

**Compile normalement. On fait appel a la méthode contains() pour vérifier qu’un point est contenu dans un rectangle. Mais comme dis précédemment pour qu’il puisse marcher normalement, il faudrait corriger la création du rectangle t ;**

* + **Systm.out.println(s.surface()) ;**

**Compile normalement. On fait appel a la méthode surface() pour calculer la surface d’un rectangle SlantedRectangle s ;**

* + **s.rotate(2) ;**

**Compile normalement. On fait appel à la méthode rotate() ;**

* + **System.out.println(s.cotains(p)) ;**

**Appel Rectangle.contains() pour vérifier que SlantedRectangle s contient un point**

**Exercice 5**

**Enoncer**

Disons si la classe Dessin définie précédemment peut contenir des rectangles inclinés et si les méthodes surface, contains et hull de la classe Dessin fonctionnent encore correctement ?

**Solution**

La class Dessin peut contenir des SlantedRectanle. Ainsi les méthodes surface, contains et hull fonctionneront toujours correctement, car elles sont definies dans la class rectangles et d’après l’exercice 3 ces méthodes ont été redéfinies dans la class SlantedRectangle. Donc en conclusion peuvent être utilisées par les objets de type SlantedRectangle.

**Exercice 6**

**Enoncer**

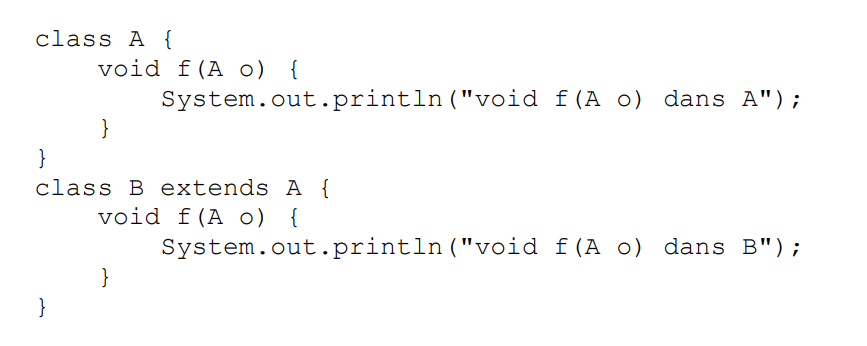
Définir une méthode String toString() dans la classe Rectangle ? Est-ce en fait une définition ou une redéfinition ? Est-il nécessaire de la redéfinir dans la classe SlantedRectangle ?

**Solution**

**Exercice 8**

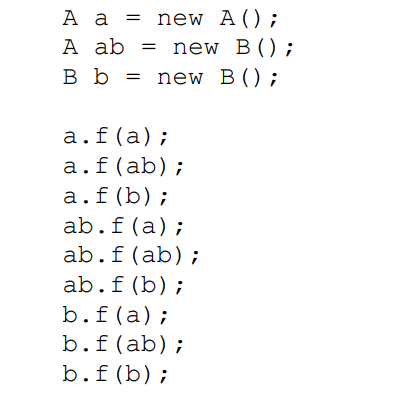
**Enoncer**

On considere les definitions de class suivantes :



**Consigne :**

Disons ce qu’affiche ce code



Apres avoir compile ce code ci-dessus en créant une class A, une class B et une classe Main nous avons eu le résultat suivant :

void f(A o) dans A

void f(A o) dans A

void f(A o) dans A

void f(A o) dans B

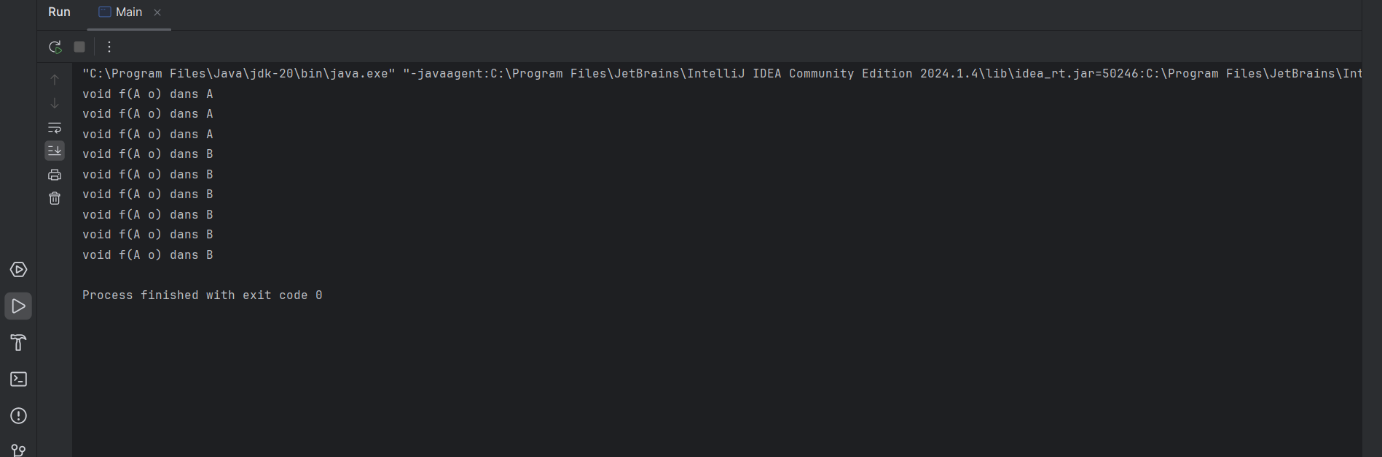
void f(A o) dans B

void f(A o) dans B

void f(A o) dans B

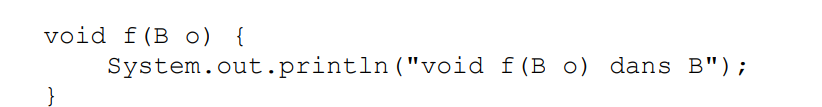
void f(A o) dans B

void f(A o) dans B



**Exercice 9**

En ajoutant la methode suivante :



dans la class B S’agit -il d’une redefinition ou d’une surcharge ? quaffiche alors le fragment du programme de l’exercice 8

**Solution**

* 1. La methode void f(B o) dans la class B est une redefinition . le programme affichera :

void f(A o) dans A

void f(A o) dans A

void f(A o) dans A

void f(A o) dans B

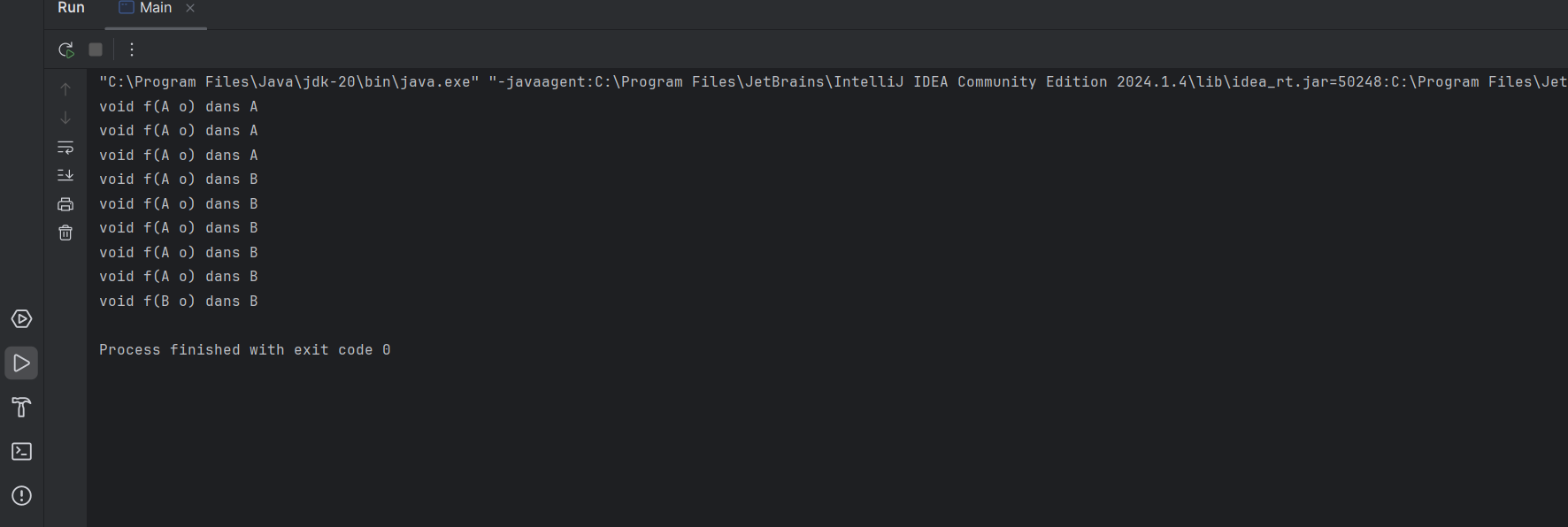
void f(A o) dans B

void f(A o) dans B

void f(A o) dans B

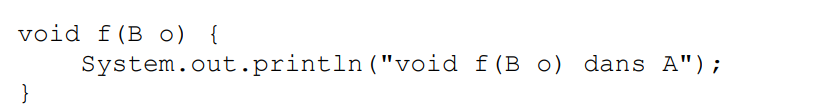
void f(A o) dans B

void f(B o) dans B



**Exercice 10**

On ajoute finalement a la class A la methode suivante :



**Consigne :**

Est-ce une redéfinition ou une surcharge ? Qu'affiche alors le fragment de programme de l'exercice 8 ?

**Solution :**

1. En ajoutant a la class A la méthode void f(B o) nous faisons une surcharge ;
2. Le fragment du code affichera :

void f(A o) dans A

void f(A o) dans A

void f(A o) dans A

void f(A o) dans B

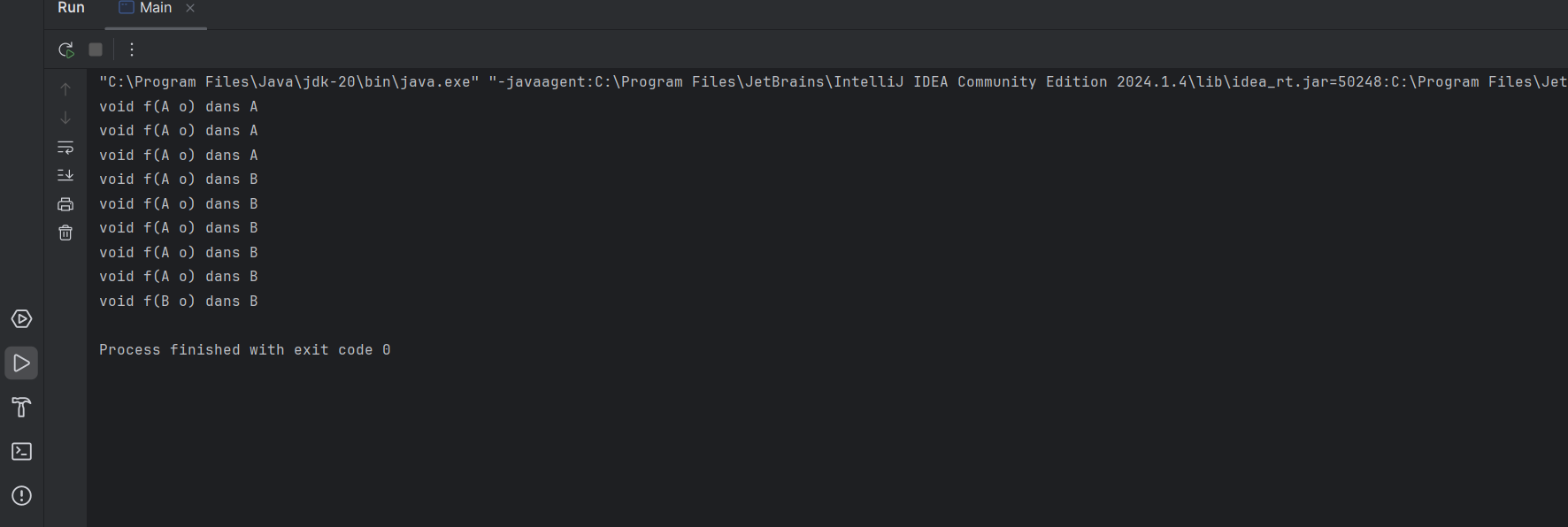
void f(A o) dans B

void f(A o) dans B

void f(A o) dans B

void f(A o) dans B

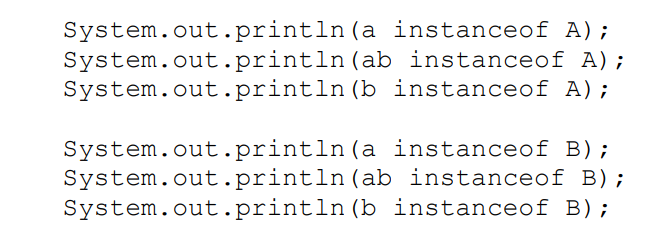
void f(B o) dans B



**Exercice 11 :**

**Consigne :**

**Qu’affiche le code suivant :**

****

**Solution**

Apres ajout de ce code dans la classe main nous avons ce resultat :

