EXERCICE 4

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

```
Point p = new Point(1,2);
Rectangle r = new Rectangle(p, 2, 3);
Rectangle t = new SlantedRectangle(p, 2, 3);
SlantedRectangle s = new SlantedRectangle(p, 2, 3);
System.out.println(r.surface()); //
r.rotate(2);
System.out.println(r.contains(p));
System.out.println(t.surface());
t.rotate(2);
System.out.println(t.contains(p));
System.out.println(s.surface());
s.rotate(2);
System.out.println(s.contains(p));
```

CODE	COMPILATION	METHODE APPELE
System.out.println(r.surface());	oui	Rectangle.surface
r.rotate(2);	non	Rotate absente dans rectangle
System.out.println(r.contains(p));	oui	Rectangle.contains
System.out.println(t.surface());	oui	Rectangle.surface
t.rotate(2);	non	Rotate absente dans rectangle
System.out.println(t.contains(p));	oui	Rectangle.contains
System.out.println(s.surface());	oui	SlantedRectangle.surface
s.rotate(2);	oui	SlantedRectangle.rotate
System.out.println(s.contains(p));	oui	SlantedRectangle.contains

EXERCICE Nº 5

Est-ce que la classe Dessin définie précédemment peut contenir des rectangles inclinés ? Est-ce que les méthodes surface, contains et hull de la classe Dessin fonctionnent encore correctement ?

// Les méthodes contains et hull ne fonctionne plus correctement et nécessitent une redéfinition tandis que la méthode surface fonctionne correctement

EXERCICE NO 8

Qu'affiche le fragment de programme suivant ?

```
A a = new A();
A ab = new B();
B b = new B();
a.f(a);
a.f(ab);
a.f(b);
ab.f(a);
ab.f(ab);
ab.f(b);
b.f(a);
b.f(ab);
b.f(b);
```

// Apres compilation le résultat que nous avons est le 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 N^{O} 9

On ajoute maintenant à la classe B la méthode suivante

void f(B o) {

System.out.println("void f(B o) dans B");

}

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

// L'ajout de la méthode `f (B o)` dans la classe `B` constitue une surcharge car elle a le même nom que f (A o) mais avec une signature différente.

Ce que l'on obtient a l'affichage après compilation

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 N^{O} 10

Exercice n° 10 On ajoute finalement à la classe A la méthode suivante

void f(B o)

{ System.out.println("void f(B o) dans A");

}

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

// L'ajout de la méthode `f (B o)` dans la classe `A` constitue une surcharge car elle a le même nom que f (A o) mais avec une signature différente.

Ce que l'on obtient a l'affichage après compilation

void f(A o) dans A

void f(A o) dans A

void f(B o) dans A

void f(A o) dans B

void f(A o) dans B

void f(B o) dans A

void f(A o) dans B

void f(A o) dans B

void f(B o) dans A

EXERCICE Nº 11

Qu'affiche le fragment de programme suivant ?

```
System.out.println(a instanceof A);
System.out.println(ab instanceof A);
 System.out.println(b instanceof A);
System.out.println(a instanceof B);
System.out.println(ab instanceof B);
 System.out.println(b instanceof B);
```

// Apres compilation on obtient

True //car a est une instance de de A

True // car ab est une instance de B et B est une sous classe de A

True // car ab est une instance de B et B est une sous classe de A

False // car a est une instance de A

True // ab est une instance de B

True // b est une instance de B

EXERCICE Nº 13

On considère les définitions de classes suivantes

```
 class C {
 char ch = 'C';
```

```
char getCh() { return ch; }
 }
 class D extends C {
char ch = 'D';
char getCh() { return ch; }

 }
```

Qu'affiche le fragment de programme suivant ?

```
C c = new C();
 C cd = new D();
 D d = new D();
System.out.println(c.ch);
System.out.println(c.getCh());
System.out.println(cd.ch);
System.out.println(cd.getCh());
System.out.println(d.ch);
System.out.println(d.getCh());
// Apres compilation on obtient :
C
C
C
D
D
D
```