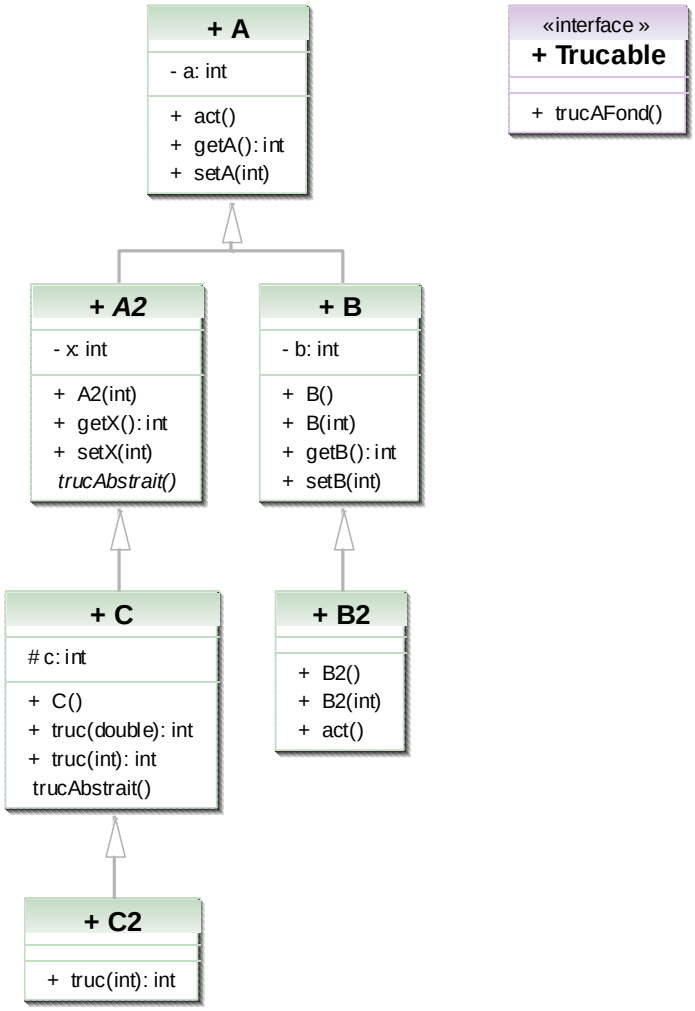
INF250 TD Programmation Correction

Code Source

```
public interface Trucable {
	void trucAFond();
}

public class A {
	private int a;

	public void act(){
		this.a++;}
	}
	public int getA() {
		return a;}

	public void setA(int a) {
		this.a = a;}
}

public abstract class A2 extends A {
	private int x;
	public A2(int x) {
		super();
		this.x = x;}

	public int getX() {
		return x;}

	public void setX(int x) {
		this.x = x;}

	abstract void trucAbstrait();
}

public class C extends A2 {

	protected int c;
	public C(){
		super(0); }


	public int truc(double a) {
		return 1; }

	public int truc(int a) {
		return 0; }

	@Override
	void trucAbstrait() {
	}
}

public class C2 extends C {
	public int truc(int a)
	{
		return 4;
	}
}
```

Diagramme de classe UML



Quelles sont les lignes qui posent problème dans le code suivant et pourquoi :

```
public class Test {

	public static void main(String[] args) {
		A a = new A();			//1 A n'ayant pas de constructeur celui par défaut est utilisé. La variable d'instance de a de l'objet a reçoit 0.
		a.setA(10);			//2 La variable d'instance a de l'objet a reçoit 10.
		a.a = a.a+1;			//3 La variable d'instance a de l'objet a est privée et ne peut être utilisée en dehors de sa classe.
		System.out.println(a.getA());	//4 Affiche la valeur de la variable d'instance a de l'objet a
		System.out.println(a);		//5 Utilise la méthode toSting() héritée de Onject sur l'objet a
		System.out.println(a.a);	//6 La variable d'instance a de l'objet a est privée et ne peut être utilisée en dehors de sa classe.
		System.out.println(a.setA(11)); //7 la méthode void setA(int a)ne peut être affichée
	}
}
```

```
//3 a est private
//6 a est private
//7 void setA(int) void ne peut-être affiché
```

Quelles sont les déclarations incorrectes ?

```
public class Z1 implements A {

}
//A est une classe et ne peut-être implements, ce mot clef est utilisé pour
les interfaces

public class Z2 extends A {

}
//Ok la clase Z2 hérite de A

public interface Z3 extends A{

}
//L'interface ne peut hériter que d'une autre interface
public interface Z4 extends Trucable {

}
//ok
```

1 et 3

Dans le code suivant quelles sont les lignes fausses et pourquoi ?

A2 a2 ;
a2.setX(10) ;
a2 = new A2() ;

2 et 3
a2 n'est pas créé (2) et a2 ne peut-être créé car A2 est abstraite et si A2 ne l'était pas A2() n'existe pas.

Que donne le code suivant ?

```
package ds_table;

public class TestB {

	public static void main(String[] args) {

		B b = new B(10);
		System.out.println(b.getA()); //1
		b.setA(10);
		System.out.println(b.getB()); //2

	}

}
```

```
//1   0     B b = new B(10); positionne b.b et appel le constructeur sans
paramètre de A, A a un constructeur par défaut qui définit b.a à 0
//2   10
```

Dans les classes A, A2, C, C2 quelles sont les méthodes surchargées ? Quelles sont les méthodes redéfinies[1] ?

Dans C truc(int) :int et truc(double) :int sont surchargés

Dans C2 truc(int) :int est redéfini

[1] La méthode trucAbstrait() de C est dite définie.

Que donne le code suivant :

```
public class TestC2 {
	public static void main(String[] args) {
		C2 c2 = new C2();
		System.out.println(c2.getA() +" "+ c2.getX()); //1
		c2.act();
		System.out.println(c2.getA() +" "+ c2.getX()); //2
		System.out.println(c2.truc(10.0));             //3
		System.out.println(c2.truc(10));               //4
		C c = new C();
		System.out.println(c.truc(10));                //5
	}
}
```

```
//1 0 0

//2 1 0

//3 1

//4 4

//5 0
```

Comment rendre B Trucable et pas A2 ? Vous expliquerez simplement ce qu'il faut faire.

B doit implémenter l'interface Trucable.
La méthode trucAFond() doit-être codée.

Quels sont les lignes qui posent problème et pourquoi ?

```
public class TestPoly {
      public static void main(String[] args) {
            Object o = new Object();            //1
            A a = new A();                      //2
            B b = new B(10);                    //3
            B2 b2 = new B2(10);                 //4

            o=a;                                //5
            o=b2;                               //6
            b =(B) o;                           //7
            b.setB(0);                          //8

            b2 = new Object();                  //9


            B bb = new B();                     //10
            B2 b2b = (B2) bb;                   //11
      }
}
```

//9 b2 = new Object(); B2 est une classe fille de Object donc beaucoup plus riche
//11 B2 b2b = (B2) bb; bb est de type B donc une classe mère de B2 donc moins riche