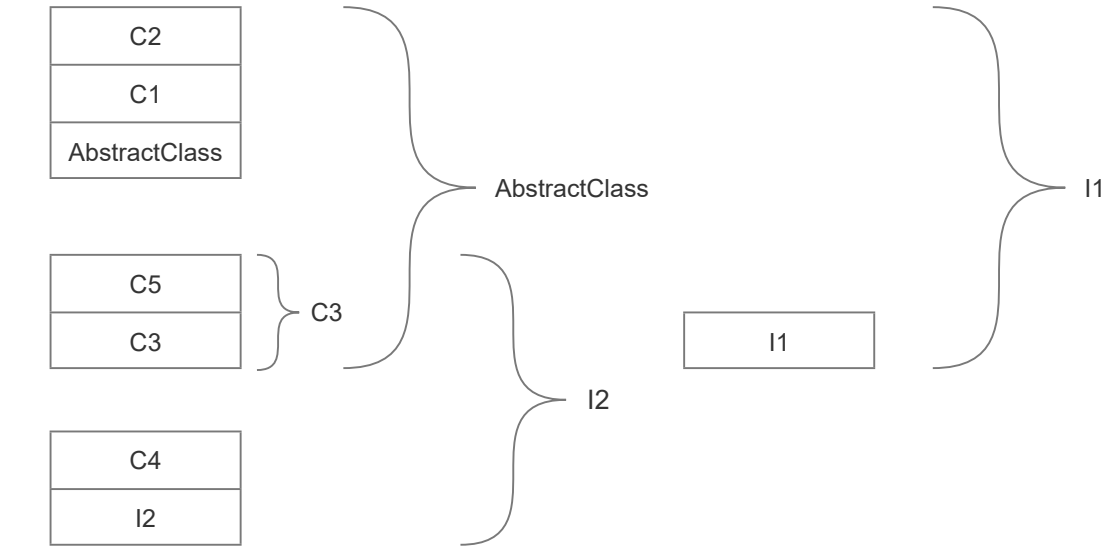
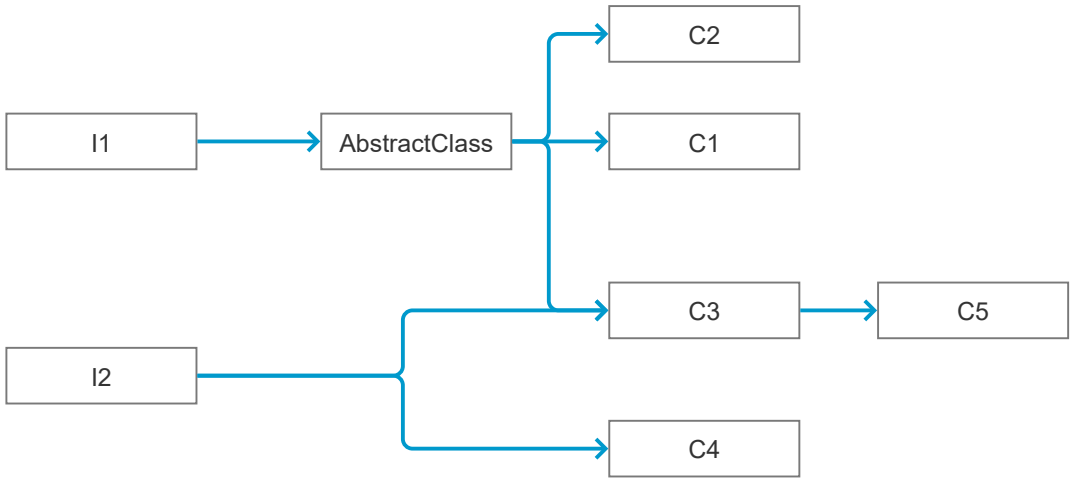
1. draw the class diagram, including interface





2. why are these declararions legal?

An interface can't be instantiated. In order to do so, we need to use a class implementing it and to instantisate the class. This process is quite like that a superclass's reference is pointing to its subclass's instance, that is, the subclass object's memory address. In short, an interface's reference pointing to its instantisating class object is just an practical example of polymorphism in Java.

3. which methods will be called and why?

3.1 anI1.i1method();
Which methods will be called: call i1method() in C1;
Why: anI1 is I1's reference instantiating via class C1 and is pointing to the memory address of object created from class C1. So this statement will call i1method() in C1.

3.2 anI1b.i1method();
Which methods will be called: call i1method() in C4;
Why: anI1b is I1's reference instantiating via class C4 and is pointing to the memory address of object created from class C4. So this statement will call i1method() in C4.

3.3 anI1b.i1and2method();
Which methods will be called: call i1and2method() in C4;
Why: anI1b is I1's reference instantiating via class C4 and is pointing to the memory address of object created from class C4. So this statement will call i1and2method() in C4.

3.4 anI1b.i1method();
The same reason as 3.2

3.5 anI2.i2method();
Which methods will be called: call i2method() in C3;
Why: anI2 is I2's reference instantiating via class C3 and is pointing to the memory address of object created from class C3. So this statement will call i2method() in C3 .

3.6 anI2.i1and2method();
Which methods will be called: call i1and2method() in C3;
Why: anI2 is I2's reference instantiating via class C3 and is pointing to the memory address of object created from class C3. So this statement will call i1and2method() in C3 .

4. which methods will be called and why?

```
C3 aC3 = new C3();
C5 aC5 = new C5();
C3 aaC3 = (C3)aC5;
```

4.1 aC3.c3andC5m();
Which methods will be called: call c3andC5m() in C3;
Why: aC3 is C3's reference instantiating via its own class C3 and is pointing to the memory address of object created from class C3. So this statement will call c3andC5m() in C3 .

4.2 aC5.c3andC5m();
Which methods will be called: call c3andC5m() in C5;
Why: aC5 is C5's reference instantiating via its own class C5 and is pointing to the memory address of object created from class C5. So this statement will call c3andC5m() in C5 .

4.3 aaC3.c3andC5m();
Which methods will be called: call c3andC5m() in C5;
Why: aaC3 is C3's reference instantiating via its subclass C5 and is pointing to the memory address of object created from class C5. So this statement will call c3andC5m() in C5 .

4.4 aaC3.c3andC5
Which variable will be called: call c3andC5 in C3;
Why: aaC3 is C3's reference instantiating via its subclass C5 and is pointing to the memory address of object created from class C5. This statement will call a variable not method and then it will call its own variable, which is 42, in C3 not in C5

4.5 aaC3.c3andC5 = 99999
The same reason as 4.4, so this statement will set c3andC5 in C3 to 99999

4.6 aaC3.c3andC5m();
The same reason as 4.3, and because of 4.5, the printing output remains the same as before.

4.5 aaC3.c3andC5
Due to 4.5, this statement will print 99999 for c3andC5, rather than 42 in 4.4

5. give an example when you would use an abstract class but not an interface

We define a abstract class - Electronics, which stands for all electronic decives. From what we know about them, all electronic decives have turning on and turning off method, but the method of a certain decive differs from that of the other. And turning on and turning off are the abstract description of electronic devices, not their behavior. So in this case, we should utilize an abstract class to do so, instead of an abstract interface.

6. give an example when you would use an interface but not an abstract class

It is easy to figure out that aiplanes and birds have the same feature - the ability to fly, but they are different things, in other words, they are not from the same class. So in order to define their common method, flying, we would use an interface and let them to implement it according to their own needs. The reason to do so is very simple. Assume we define flying as a class method, that is, all subclasses that inherit the superclass would all have to have this method or to implement it. But in reality, not all birds can fly, for examle, chickens or ostriches. Therefore, it is not appropriate that we define the flying as a method of an abstract class.

7. give an example when you have to use an interface

We have to use an interface when some unrelated classes have commond features to implement . For example, LinkedList has dozens of unique features, such as deque and queue, which other lists don't have like ArrayList. If we define those methods in the super class, AbstractList, it means all subclasses should have those methods to inherit. However, note that other unrelated classes might use those methods, like ArrayDeque. In consequence, we have to define those methods as an interface.