// why are these declarations legal?

4). Class C1 implements the interface I1 and therefore a reference variable of type I1 can hold the reference of an object of type C1.

5). Class C3 implements the interface I2 and therefore a reference variable of type I2 can hold the reference of an object of type C3.

6). Class C2 implements the interface I1 and therefore a reference variable of type I1 can hold the reference of an object of type C2.

7). Class C4 implements the interface I1 and therefore a reference variable of type I1 can hold the reference of an object of type C4.

8). Class C4 implements the interface I2 and therefore a reference variable of type I2 can hold the reference of an object of type C4.

// which methods will be called and why?

12). anI1 contains the reference of an object of class C1. Therefore, due to dynamic method dispatch, i1method() defined in class C1 will be called.

13). anI1b contains the reference of an object of class C4. C4 overrides the i1method(). Therefore, due to dynamic method dispatch, i1method() defined in class C4 will be called.

14). anI1b contains the reference of an object of class C4. C4 overrides the i1and2method(). Therefore, due to dynamic method dispatch, i1and2method() defined in class C4 will be called.

15). anI1b contains the reference of an object of class C4. C4 overrides the i1method(). Therefore, due to dynamic method dispatch, i1method() defined in

class C4 will be called.

16). anI2 contains the reference of an object of class C3. C3 overrides the i2method(). Therefore, due to dynamic method dispatch, i1method() defined in class C3 will be called.

17). anI2 contains the reference of an object of class C3. C3 overrides the i1and2method(). Therefore, due to dynamic method dispatch, i1and2method() defined in class C3 will be called.

// which methods will be called and why?

24). aC3 contains the reference of an object of class C3. Therefore, c3andC5m() defined in class C3 will be called.

25). aC5 contains the reference of an object of class C5. Therefore, c3andC5m() defined in class C5 will be called.

26). aaC3 contains the reference of the C5 class object typecasted into class C3. Though typecasted, it still refers to a C5 class object and therefore, due to dynamic method dispatch, c3andC5m() defined in class C5 will be called.

27). aaC3 is an object of type C3 and therefore the value of c3andC5 contained in the class C3 will be printed.

28). aaC3 is an object of type C3 and the value of its instance variable c3andC5 defined in class C3 is changed to 99999.

29). aaC3 contains the reference of the C5 class object typecasted into class C3. Though typecasted, it still refers to a C5 class object and therefore, due to dynamic method dispatch, c3andC5m() defined in class C5 will be called.

30). aaC3 is an object of type C3 and therefore the value of c3andC5 contained in the class C3 will be printed.

****** // give an example when you would use an abstract class but not an interface *******

I would use an abstract class when I need instance variables that are not public or static or final. Also, when there is close resemblance between the superclass and subclass. For example, if I have to initialize some instance variables to perform a calculation and want the subclass to inherit it then abstract class should be used.

```
abstract class AbstractClass1 {

	int x;
	int y;

	void set(int val) {
		x = 5;
		y = x * 2;
	}

	public abstract void abstractClass2();
}

public class AbstractClass extends AbstractClass1 {

	public void abstractClass2() {
		System.out.println("AbstractClass");
	}

	public static void main(String args[]) {
		AbstractClass a = new AbstractClass();
		a.set(3);
		System.out.println(a.x);
		System.out.println(a.y);
	}
```

```
}
```

******* // give an example when you have to use an interface
// give an example when you would use an interface but not an abstract class ********

Interfaces can be used when multiple standalone functionalities are to be implemented. Therefore, if there is a class Audio and class Video and I want to make a class SmartWatch that contains both audio and video then I cannot use an abstract class since functionality of both audio and video classes is required into the SmartWatch class but multiple inheritance is not supported by java.
But, a class can implement multiple interfaces and therefore interfaces are a suitable choice in such cases.

```
public interface I1 {
	default void produceAudio() {
		System.out.println("Audio produced");
	}
}

public interface I2 {
	default void produceVideo() {
		System.out.println("Video produced");
	}
}

public class SmartWatch implements I1, I2 {

	public static void main(String args[]) {
		SmartWatch s = new SmartWatch();
		s.produceAudio();
		s.produceVideo();
	}

}
```