1. Any subclass that is going to inherit from either an abstract class or an interface is going to have to implement some type of method. The methods it's going to have to impliment is called abstract methods. An abstract method is a method without a body.

An abstract class usually contains at least one abstract method that cannot be instantiated. It can also have no methods at all. An abstract class can have normal methods and abstract methods in it.

An interface contains only abstract methods that cannot be instantiated either.

Example. If we have singers. We can have an abstract class called singers.

The class can then have an abstract method such as public abstract void stageCostume(); and a normal class called public void paymentForSinging(int hours) { system.out.println("The singer will make $" + 100 + " by singing tonight"); }

The abstract method can then be extended to a sub class for let us say, singer 'Lisa' and 'Bob' where it then can be filled out specifying what type of stageCostume the singers like to use.

An interface will ONLY have abstract methods without bodies. Every method is bodyless.

It is good to use an interface when the methods your subclasses use are going to be unique to each class.

You could then put in public abstract void paymentForSinging(int hours); as a class, and then you can fill in the payment they will get in their own class.

2. The downside of inheritance is that you cannot use multiple inheritances for subclasses.

3. To get a customer to describe a system he wants, I would ask them how they imagine it should work. What features should it have, what should it/should not do. Make them describe the system as if it was already made and did exactly what they wanted to do.

4. Coding is a part of software development. Writing code is making instructions that a computer will understand. Software development is the full package of a product. It will contain both design, maintenance, testing and much more.

5. Good code is easy to read and as simple as possible. No need to make the code longer or more complex than it must be. If you can make it easy to read and functional with 20 lines of code instead of 40, then do so.