**OOP Master🧙🧙‍♂️**

What is the difference between an interface and an abstract class? Write down some examples where you would use an interface and some examples where you would use an abstract class

* The abstract class, is used to inherit certain attributes and methods for subclasses, but the abstract class itself cannot be instantiated.  
  An interface is similar to an abstract class in that it cannot be instantiated, but while the abstract class can contain both attributes and methods, the interface works like a set of rules when implemented in a class – It dictates which methods a class must have. Furthermore, a class can implement infinite interfaces but can only inherit from a single abstract class.  
  Interfaces can be used for example when creating NPC’s in a videogame. For the NPC’s to work properly they all need some of the same methods e.g. speak(), walk() etc.  
  NPC’s however do not always have the same attributes. This is where abstract classes come in handy. While some might be of race ‘Tauren’, which maybe has an attribute called ‘horns’, the race ‘Night Elf’ does not have this attribute, but both of them need the methods required for them to function as an NPC.

What are the downsides of inheritance?

* If u have a superclass called ‘Character’, it might have some attributes called ‘legs’, ‘arms’, ‘hair’ and ‘name’. This can be a problem if u choose to create a new character of type ‘Blob’, this class might not need the attributes ‘legs’, ‘arms’ & ‘hair’ which is a downside.

How do i get from a customer describing a system he wants to have to a fully coded system? What steps would you take?

* Firstly, you ask a lot of questions regarding the system wanted by the customer. It is important to make sure that the project is specified fully. From there, you can make a domain model, and go it over with the customer and make sure that you both agree on the design. From there on, pseudo code and class diagrams, can help you further sketch out the system, before finally starting coding it.

What is the difference between coding and developing software?

* Coding is a small part of developing software, however a small one. Developing software is about talking to the customer, finding out the requirements, working together with colleagues, continuously consulting the customer and making sure the system is meeting all expectations.

What do you think is good code?

* Good code is short, precise, is not redundant, commented, if necessary, has good variable names, user-friendly and generally has good functionality. Good code needs to be thoroughly tested.