



OOP – Inheritance

Model Answer Approach

[](http://www.hyperiondev.com/portal/)

# Auto-graded task 1

This solution takes the approach of creating each of the classes, attributes, and methods as detailed within the task instructions. It starts with the Course class, incorporating a class attribute for the head office location and a method to display this location.

Following this, the OOPCourse class is created with its constructor initialised with default values for the description and trainer attributes. The required methods within the OOPCourse class are then created to handle the required functionality of each method.

This approach is suitable, as it demonstrates the basics of inheritance by inheriting class attributes and methods from the Course class, providing a solid introduction to single inheritance.

One challenge you may encounter is defining default values for the attributes within the OOPCourse class. However, as there are multiple approaches, it presents a good opportunity to experiment with different methods of setting default values for attributes.

# Auto-graded task 2

For this task, the approach which has been taken is to begin by creating the required Adult class. This enables the attributes to be set along with a starting method called can\_drive, which is responsible for displaying that the created object(person) is old enough to drive.

Subsequently, the Child class can then be created and inherit all the attributes and methods from the Adult class. Through the use of method overriding, it is possible to override the inherited can\_drive method with an altered version, which displays that the created object(person) is not old enough to drive. In this case, there is no need to create a new constructor method, as no new attributes are being introduced within the Child class.

Within the logic after the the classes are defined, the user is then prompted to enter the values which are required when creating an instance of the Adult or Child class. The provided age is then used to determine if an instance of the Adult or Child class should be created based on the age of the user. This makes it possible to call the can\_drive method only once, as its output will be determined based on whether the object is an instance of the Adult or Child class.

This is a fitting approach, as it demonstrates that subclasses like Child do not always need a constructor when no new attributes are being added to the constructor. Additionally, it demonstrates the ability to use conditional statements to control which class is used to create the object.

A common pitfall to avoid is mistakenly overriding the Adult class constructor in Child without introducing any new attributes.