Object-Oriented Programming in Kotlin

This is a technique used to break down a complex problem into smaller bits by creating objects. In OOP we structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects. Objects have state and behaviour e.g

A car is an object. Its state includes attributes like `color`, `make`, `model`, `registration`. The behaviours of a car could be, `start()` , `stop()` , `accelerate()` , `hoot()` etc

Class

A class is a template/ blueprint for an object. Before an object is created a class must be defined. One class can be used to create multiple objects. Since a class is a template for an object we can use this analogy:
Before a house is constructed, an architect draws up a plan with details about where the doors, windows, rooms etc will be and all their measurements. This one plan can then be used to construct many identical houses. In this case the plan is the class and the houses are the objects.

Creating a Class

```
class Car(var make: String, var model: String, var
registration:  String, var speed: Int) {

 public fun start() {
 println("I am starting")
 }

 public fun accelerate(acceleration: Int): Int {
 speed = speed + acceleration
 return speed
 }

}

fun main() {
 var gari = Car("Nissan", "Leaf", "KDA 379G", 0)
 gari.start()
 println(gari.speed)
 gari.accelerate(30)
 println(gari.speed)
}
```

A class is created using the `class` keyword. Next we have the class name and parameters that make up its primary constructor. A class may be created without parameters in the primary constructor. If parameters are specified, they must be provided when creating an instance of the class. The body of the class is demarcated by a pair opening and closing curly braces `{ }`.

From the Car class we are able to create an object and assign it to a variable `gari`. We access the object's properties (state) and functions (behaviour) using the dot `.` notation.

```
println(gari.registration) //prints out KDA 397G
println(gari.make) //prints out Nissan
```

Complete the class above by adding `deccelerate()`,`hoot()` and `stop()` methods

Data Class

Data classes are designed to hold data. A data class is created using the data keyword.

```
data class Book(var title: String, var author: String, var pages: Int)

fun main() {
 var book = Book("Coming To Birth", "Marjorie Macgoye", 172)
println(book) //Book(title=Coming To Birth, author=Marjorie
Macgoye, pages=172)
 println(book.title) //Coming To Birth
}
```

The data class primary constructor must have at least one parameter. All primary constructor parameters must be marked either `var` or `val` .