**Creating a Class**

Creating a class in C++ is again very similar to Java and C#. Classes are very good for describing objects in your applications, providing a template with member variables and functions. Below is a class that describes a virtual pet; it knows how hungry a pet is and its name.



Creating a new instance of a class in C++ is identical to Java. The only thing to consider is that the definition of the class (code above) must be before the declaration. The compiler needs to know what the class is before it can create an instance. The following will create a single instance of the CyberPet class:



**Access Modifiers**

Just like in Java and C#, C++ has three access modifiers to allowing the programmer to decide how elements of their class can be accessed. Public states the item can be accessed by elements outside the class, such as the main. Private states the item can only be accessed by that instance of the class; nothing else can see or touch a private variable. Protected is the same as private but also allows classes that inherit from the class to access. We will cover inheritance later on in this worksheet.

By default, all items in a class are private, whereas all items in a struct are public.

As seen by the example above, rather than the access modifier for each function and variable, we allocate sections of the class. You can have these in any order and have them multiple times.



**Inheritance**

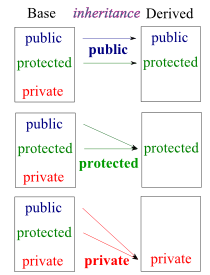
Inheritance in C++ is done during the declaration of the class. You can create a class that inherits from any other (there are exceptions) which allows you to extend what they can do. The syntax for inheritance is as follows:



Remember that class A has to be defined before class B or else the code will fail to compile. Sometimes this is impossible, so we can use a forward declaration to state a class exists before we have defined it:



You may also notice that we are using a public inheritance. We can inherit using any of the access modifiers but it changes how we can access and use the items we are inheriting. The following diagram shows what happens to each element in class when it is inherited using the different modifiers:

As private variables are always unique to that class, the derived class can never access them. The access modifier converts either the public and protected modifier. Public keeps them the same, protected converts public to protected in the derived class, and private converts both public and protected to private items in the derived class.

**Custom Constructors**

Often you will create a class with a custom constructor. You can create as many constructors as you want for a class and the compiler will know which one to choose based on the arguments you give it.



**Exercises**

1. Add a Getter and Setter for the name of the pet in the class.
2. Create three instances of CyberPet in the main and give each a name and a hunger value.
3. Print out the names and hungers of each CyberPet.
4. Add a variable that describes how happy the CyberPet is.
5. Create a function inside the CyberPet class that prints out the name, hunger and happiness.

**Summary**

* Unlike Java where you can set each variable and functions as private, public and protected, in C++ we usually split the related items into sections.
* Constructors do not return a type, but they can take arguments if designed to do so. A constructor that takes no arguments is called the ‘default constructor’.
* There are three access modifiers; public, private and protected.
* You can forward declare a class if you want to use it before defining it.
* The access modifier we use to inherit will affect how public and protected variables are treated by the derived class.