# IN710 OOSD 2016 Practical 4.1 – Game Character Class

In this practical, we will use the C# **interface construct** and the **Strategy Design Pattern** to build the beginnings of a Game Character class. The general behaviour is described below; you will need to figure out how, using interface(s) and the Strategy Design Pattern, to produce an elegant, extensible implementation. There are of course, a variety of ways one might design the architecture for this application. In this exercise, **it is a requirement** that you use the Strategy Design Pattern.

There is a demo showing the required functionality on the I: drive. Please note that in the demo, you start with an initial set of game characters. This is just to allow use of the program without requiring the user to first create characters. I suggest you implement this in your own application.

# Description

1. Your application must allow the user to create characters, specifying name and kind (see Requirements below for details).
2. Your application must allow the user to select two of the created characters for “battle”. Battle in this case is simply announcing their name and rank, and describing the use of their weapon as detailed below.
3. Your application must allow the user to change a character’s weapon as desired at runtime.

# Requirements

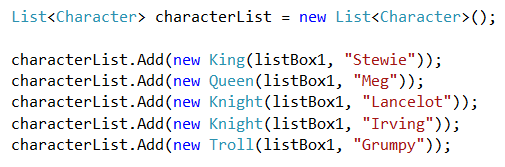
1. Your game has (at least) four kinds of characters: King, Queen, Knight and Troll.
2. Each character has a name (e.g. King Henry VIII), which is provided to it at construction.
3. Each character has a listBox that it uses for display, which is provided to it at construction or passed in when needed, whichever style you prefer.
4. Each character type has a default weapon. Kings start with a sword, Knights with a knife, Queens and Trolls with a bow.
5. It must be possible to change a Character's weapon at runtime.
6. Each weapon has a particular action -- swords slice, knives stab and bows shoot. These action words are important when weapons are 'used in battle' (see item 7c, below).
7. Characters have the following behaviours (using the listBox for display):
   1. Characters can declaim their rank/type. Each type of character does this differently, as follows:

|  |  |
| --- | --- |
| **Character Type** | **Says** |
| King | I am the most mighty of Kings! |
| Queen | I am the powerful Queen! |
| Knight | I am a chivalrous Knight! |
| Troll | Trolls don't have time to chat! |

* 1. Characters can state their name. All types of character do this in the same way, saying "My name is " followed by their name.
  2. Characters can use a weapon. In a real game, this would be part of a potentially complex fighting algorithm, which would be different for each kind of weapon. In this simplified implementation (as in the demo), weapon use simply involves explaining the weapon action. For example, ‘I shoot with my bow!’ or ‘I slice with my sword!’. Think very carefully about which class should implement this behaviour. Whose responsibility should it be?

# Code Fragments:

# Creating dummy data (in the Form, or in a manager class):



# Useful Snippet:

public interface IWeaponBehaviour

{

string UseWeapon();

}

# Planning

1. Before you begin coding, fill out the following table. In the left-hand column are entities from the Employee example we discussed in lecture. In the right-hand column you are to write the corresponding entity from the current task (if there is more than one option – e.g. King, or Queen or Knight or Troll – you can just give one exemplar). You will need to create your own names for the items in the right-hand column.

|  |  |
| --- | --- |
| **Employee Example** | **Game Character task** |
| Employee class |  |
| ContractEmployee class |  |
| IPayComputingMachine interface |  |

1. Before you begin coding, sketch a UML-style diagram of your class architecture.
2. What unit tests might you write for this architecture?