Inheritance

Allows us to reuse & extend existing code

Take the String class for example...
Currently it is just plain text
But what if we wanted add text styling ?
(Bold, Italics, Underline etc.)

We can "extend" the basic String Class,
making use of all the existing methods,
but also adding in some of our own...

StyledString Class

```
class StyledString extends String
{
    void setUnderlined(boolean underline)
        // Some code in here !
    }

    void setItalics(boolean italic)
        // Some code in here !
    }
}
```

What we get

We have only added two new methods:
- setUnderlined
- setItalics

But because we have extended String, we get:
- length
- charAt
- contains
- toLowerCase
- etc.

All for free !

Using StyledString

```
public static void main(String args[])
{
    StyledString message = new StyledString("Hello");
    message.setUnderlined(true);
    message.setItalics(true);
    System.out.println(message);
}
```

run
StyledString

Overriding

Adding features to a child (the extending class)

Is often achieved by replacing existing methods

With the new one containing additional features

This is called...

Overriding

For example...

ALL classes have a method to get text for printing
We can "override" this in the StyledString class
And include special tags for styling:

```
"\033[4m" + text + "\033[0m";
```

The above is underlining
Don't worry - these codes aren't part of this unit !
(You don't need to know or understand them)

Method Chaining

If we are really clever...
We can reuse the method of super (parent) class
And then tack on the extra features at the end:

```
public String toString()
{
  String text = super.toString();
  if(isUnderlined) text = "\033[4m" + text + "\033[0m";
  if(isItalics) text = "\033[3m" + text + "\033[0m";
  return text;
}
```

Polymorphism

What if we don't know what kind of String to expect ?

It might be a plain String or maybe a StyledString
Don't want to write a different method for each type

Luckily we don't have to !
We can just write a general purpose method:

```
public void spellCheck(String text) ...
```

And this will be able to operate on ANY kind of String
(Any Object that is a String or sub-class of String)