4. **Inheritance, Abstract Classes and Interfaces**

(a) You are asked to write a series of classes for storing document information for an office suite of applications.

i. Define an abstract `Document` class to store the following attributes: filename, author, date created and date last modified. [5]

ii. Define a `WordDocument` class that builds upon your previous class but includes the following additional `String` attributes: title, subtitle, body, header and footer. [4]

iii. Define a `SpreadsheetDocument` class that extends the `Document` class and stores a title and a two dimensional array of `Cell` objects.
The `Cell` class is defined like so: [3]

```
public class Cell {
    Object value;
    public Cell(Object value) {
        this.value = value;
    }
    public Object getValue() {
        return value;
    }
    public void setValue(Object value) {
        this.value = value;
    }
}
```

(b) Two interfaces are required in the office suite. One indicates that a document can be printed, the other indicates a document can be saved.

i. Define an interface called `Printable` that would ensure that any class that implements the `Printable` interface contains a `print` method. You may assume the `print` method has no parameters and no return value. [2]

ii. Define an interface called `Saveable` that would ensure that any class that implements the `Saveable` interface contains a `save` method. The `save` method will take a single `String` parameter that holds the filename and returns a single value indicating whether the save was successful. [2]

(c) With reference to your answers to (a) and (b), demonstrate how you would build a document class that can be saved and printed. You do not need to provide complete implementations of any method. [4]