**Writer Design**

As per the Requirement, to support multiple types of writers and support for default actions which is applicable for all types of writers, I have decided to implement polymorphism with inheritance concept using an abstract class with both concrete methods and abstract methods for writer specific implementation and default futures support.

**Writer:** is an abstract class with implementation methods such as lowercase(), uppercase(), stupidRemover() and etc for the default actions which applicable for all the subclasses by default.

Also have abstract methods such as get(), close(), write() to support individual write functionality and read functionality.

So this implementation helps to extend future writer classes and default actions can be inherited automatically. And any future actions can be added in Writer classes and without any change in implementation classes those new futures can be applied to subclasses by default as per the inheritance concept.

So this way we need not to make any changes to existing classes whenever we add new futures to all writer classes and users can choose the operations they wish to do without any dependency.

**Writer**

FileWriter

SockerWriter

Void write(String data){ - - }

Void close() { -- -- }

String get() { -- -- -- }

Writer lowercase(){ -- -- -- }

Writer uppercase() { --- -- -- }

Writer stupidRemover() { --- -- -- }

Writer duplicateRemover(){ --- -- -- }

Void write(String data){ - - }

Void close() { -- -- }

String get() { -- -- -- }

Writer lowercase(){ -- -- -- }

Writer uppercase() { --- -- -- }

Writer stupidRemover() { --- -- -- }

Writer duplicateRemover(){ --- -- -- }

StringWriter

Void write(String data){ - - }

Void close() { -- -- }

String get() { -- -- -- }

Writer lowercase(){ -- -- -- }

Writer uppercase() { --- -- -- }

Writer stupidRemover() { --- -- -- }

Writer duplicateRemover(){ --- -- -- }

Abstract void write(String data);

Abstract String get();

Abstract void close();

Writer lowercase(){ -- -- -- }

Writer uppercase() { --- -- -- }

Writer stupidRemover() { --- -- -- }

Writer duplicateRemover(){ --- -- -- }

**Void write()** : writes the content to specific writers and it’s an individual classes responsibility to implement the write logic.

**String get() :** Since this is also an abstract method and should be implemented in all the child classes to get the data from specific writers.

**Void close() :** this is to close the streaming of data to writer. Any operation made after this methods call the data is simply ignored and can’t be written to any writer.