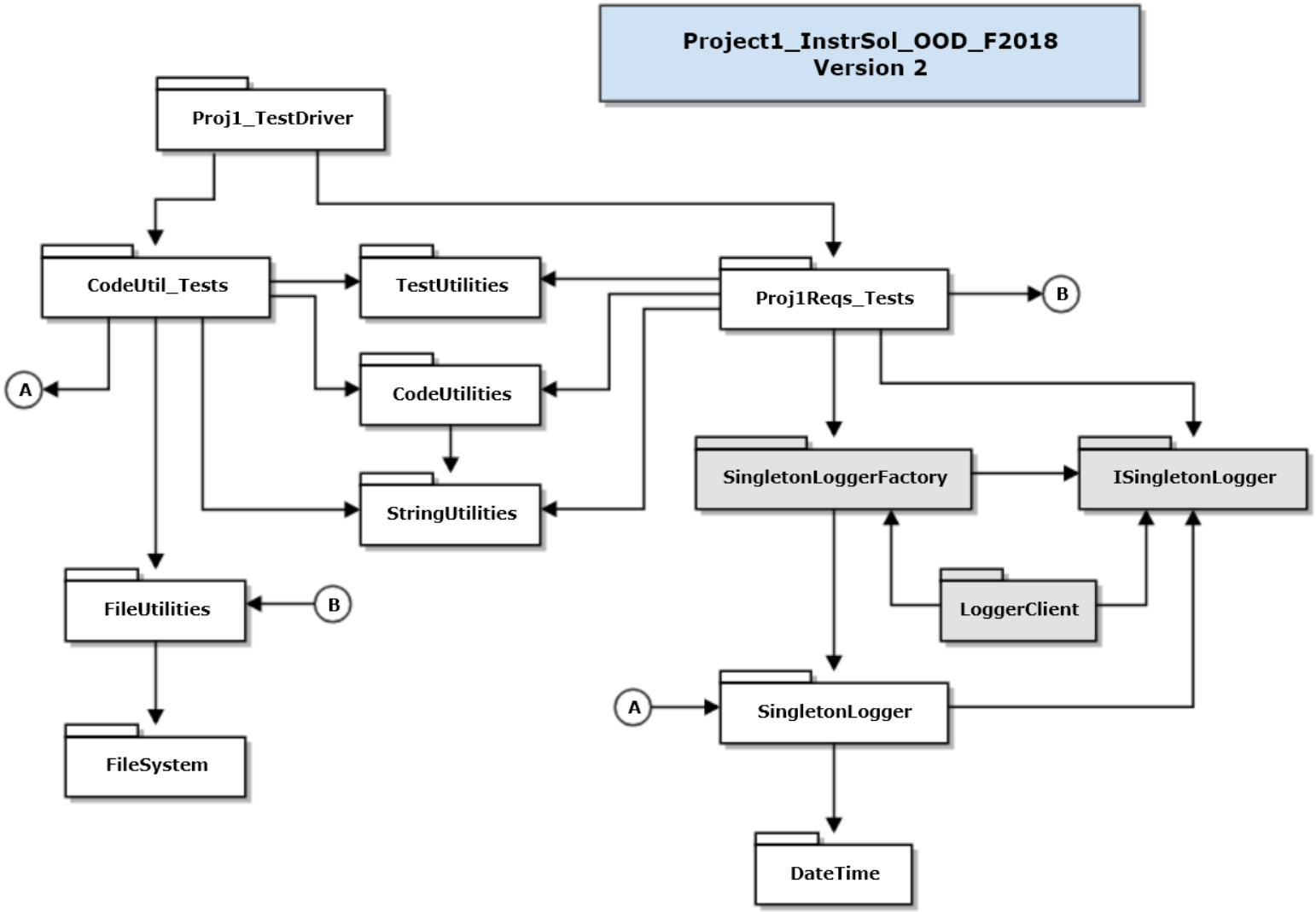
Project1_InstrSol_OOD_F2018
Version 2
Proj1_TestDriver
CodeUtil_Tests
TestUtilities
Proj1Reqs_Tests
B
A
CodeUtilities
SingletonLoggerFactory
ISingletonLogger
StringUtilities
FileUtilities
B
LoggerClient
A
SingletonLogger
FileSystem
DateTime

## ISingletonLogger.h : Logger Interface

```
template<int Category, template<int Category> class Locker>
class ILogger {
public:
  virtual ~ILogger() {}

  virtual void addStream(std::ostream* pStream) = 0;
  virtual bool usingStream(std::ostream* pStream) = 0;
  virtual bool removeStream(std::ostream* pStream) = 0;

  virtual void setTerminator(const Terminator& term) = 0;
  virtual void setAuthor(const std::string& name) = 0;

  virtual void writeHead(const std::string& msg) = 0;
  virtual void write(const std::string& text) = 0;
  virtual void writeTail(const std::string& msg = "end of log") = 0;
};
```

## SingletonLoggerFactory.h : Logger Factory Declaration

```
template<int Category, template<int Category> class Locker>
struct SingletonLoggerFactory
{
  static ILogger<Category, Locker>* getInstance();
};
```

## SingletonLoggerFactory.cpp : Logger Factory Implementations

Can’t be template because client won’t include cpp file

```
ILogger<0, Lock>* SingletonLoggerFactory<0, Lock>::getInstance()
{
  return Logger<0, Lock>::getInstance();
}

ILogger<0, NoLock>* SingletonLoggerFactory<0, NoLock>::getInstance()
{
  return Logger<0, NoLock>::getInstance();
}

ILogger<1, Lock>* SingletonLoggerFactory<1, Lock>::getInstance()
{
  return Logger<1, Lock>::getInstance();
}

ILogger<1, NoLock>* SingletonLoggerFactory<1, NoLock>::getInstance()
{
  return Logger<1, NoLock>::getInstance();
}
```