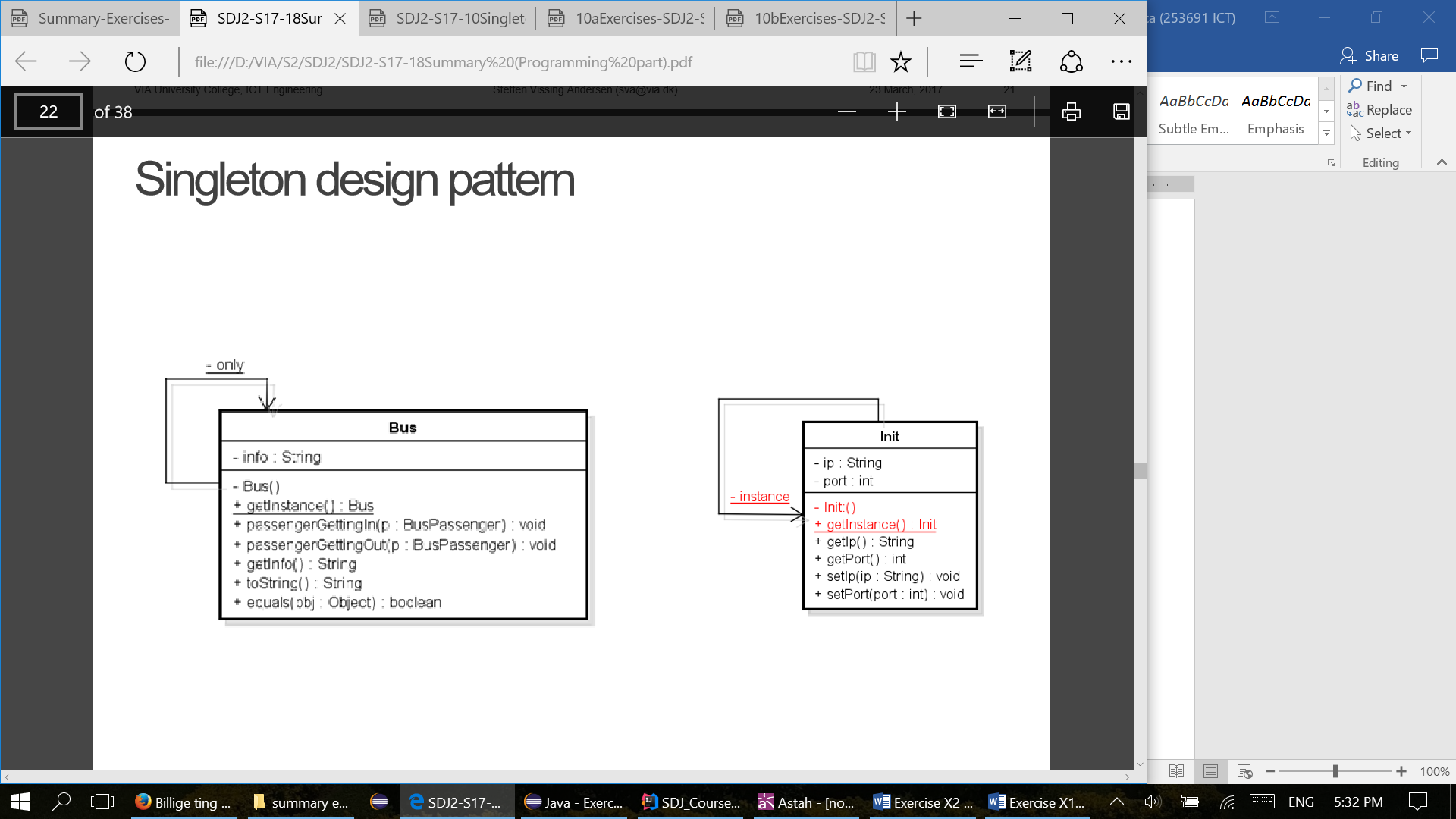
**Singleton**

**A general UML class diagram of a Singleton design pattern.**



**The overall purpose for the Singleton design pattern.**

Singleton involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

**What makes it a singleton:**

- private static Singleton instance - Private static class variable of the same type as the class that is the only instance of the class.

- private Singleton() – Private constructor to restrict instantiation of the class from other classes.

- public static Singleton getInstance() – Public static class method returning an instance of the class, this method has to ensure that only one instance is create. This is the global access point for outer world to get the instance of the singleton class.

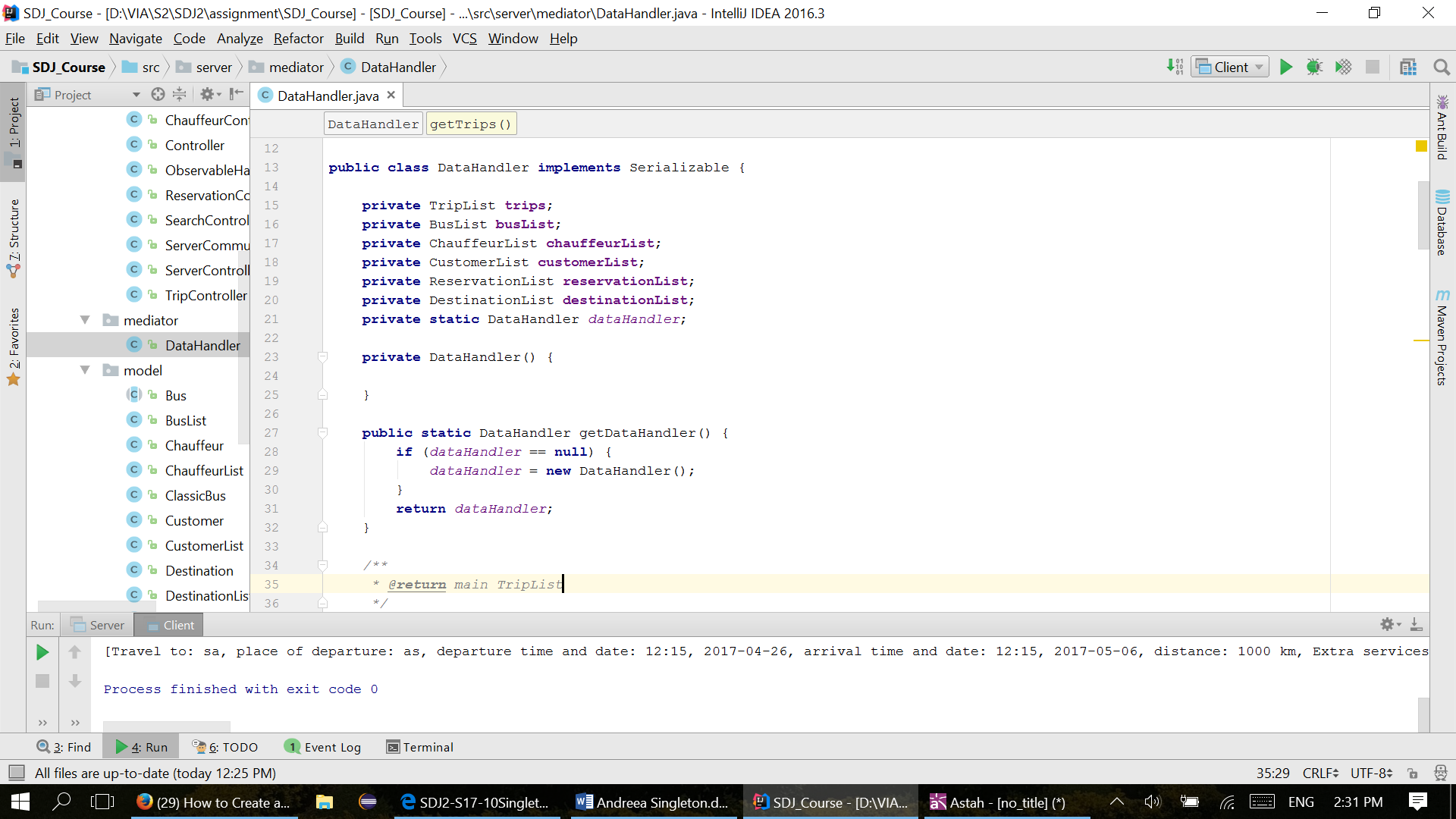
**How to use a Singleton design pattern.**

Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings etc.

To use the singleton class, we need to have static member of class, private constructor and static factory method.

* **Static member:** It gets memory only once because of static, it contains the instance of the Singleton class.
* **Private constructor:** It will prevent to instantiate the Singleton class from outside the class.
* **Static method:** This provides the global point of access to the Singleton object and returns the instance to the caller.

**Code examples for the Singleton pattern.:**



**The UML class diagram of the Singleton design implemented:**

