**Assignment 8: Design pattern, Lab assignment Custom Annotation and Reflection**

Q1. Implement singleton design pattern, to ensure that a class have only one instance and provide global point of access to it
While designing singleton pattern consider following factors, what is good progamming practice about singleton pattern?

1. Eager initialization
2. Static block initialization
3. Lazy Initialization
4. Thread Safe Singleton
5. Serialization issue
6. Cloning issue
7. Using Reflection to destroy Singleton Pattern
8. Enum Singleton

Q2. Create custom annotation to represent meta data as described:-Create annotation One is the Author Annotation and the other is the Version Annotation.

```
import java.lang.annotation.*;
@Target(value = { ElementType.CONSTRUCTOR, ElementType.METHOD, ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
public @interface Author {
	String name() default "unknown";
}
```

Notes: Note the definition of the Author annotation. This is a single-valued Annotation meaning that this Annotation has a single property called name. Also make a note of the Target Annotation. The presence of Target Annotation tells that Author Annotation can only be applied to Java elements like Constructor, Method and Type (Class/Interface). Another important thing to note is the Retention for this Annotation is set to Run-time because we want this Annotation information to be available to the JVM while the program is running. The name property is also given a default value

unknown if the consuming Application fail to provide an explicit value. Following is the definition of the Version Annotation. It looks exactly the same as

Author Annotation except the fact that it has a property called number (which is of type double) to hold the version value.

**Version.java**

```
@Target(value = { ElementType.CONSTRUCTOR, ElementType.METHOD, ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
public @interface Version {
	double number();
}
```

Let us have a look at the following class definition that makes use of the above declared Annotations. The Annotations are applied at the class-level as well as in the

```
@Author(name = "Johny")
@Version(number = 1.0)
public class AnnotatedClass {
	@Author(name = "Author1")
	@Version(number = 2.0f)
	public void annotatedMethod1() {
	}

	@Author(name = "Author2")
	@Version(number = 4.0)
	public void annotatedMethod2() {
	}
}
```

Now write an program to makes use of the new API to read the Annotation related information that are applied on various Java Elements. Write an utility method called readAnnotation() takes a parameter of type AnnotationElement which can represent a Class, method or Constructor. Then it queries for a list of Annotations of that particular element by calling the getAnnotations() method. Then the array is iterated to get the individual element, then made a downcast to extract the exact information – Author.name() and Version.number().