Annotations

1. Use java built in annotations (Deprecated, Override & SupressWarnings) in a sample class & study their behavior.
2. Create a custom annotation named @Getter. Use this annotation in a sample class for various getter methods to denote that these methods are getter methods. Write a main program using reflection to confirm that @Getter annotation is used for only getter methods. (Note that to confirm a method is getter you must have a corresponding attribute in the class)

Java Enums

1. Write anenum named ‘Status’ with the possible values NEW, REJECTED, ACCEPTED, COMPLETED. Assign a numeric value to every order status & add possibility to retrieve this value from Status object. Write a class Order that maintains order status along with order id, name, quantity, price etc.

Nested classes

1. Write an application that maintains the cache of data & helps us to retrieve inserted data using its key. The cache should also maintain private data about the timestamp when the data was added into cache. Note: refer class HashMap & add a nested class to maintain the cache data.
2. Write an application for hospital that provides an alert message when patient’s blood pressure goes above 140. Note: you will supply blood pressure as random number & you should have a system that continuously keeping watch on the blood pressure value. If it goes above 140, it should fire an event & call a callback method that simply prints alert message.