STATIC AND NON-STATIC NESTED CLASS

NESTED CLASSES

JAVA ALLOWS YOU TO DEFINE CLASSES
INSIDE OTHER CLASSES

THERE ARE SPECIFIC SITUATIONS IN WHICH
IT MAKES A LOT OF SENSE TO DO SO..

..AND SO JAVA PROVIDES A FEW DIFFERENT
WAYS OF CREATING NESTED CLASSES

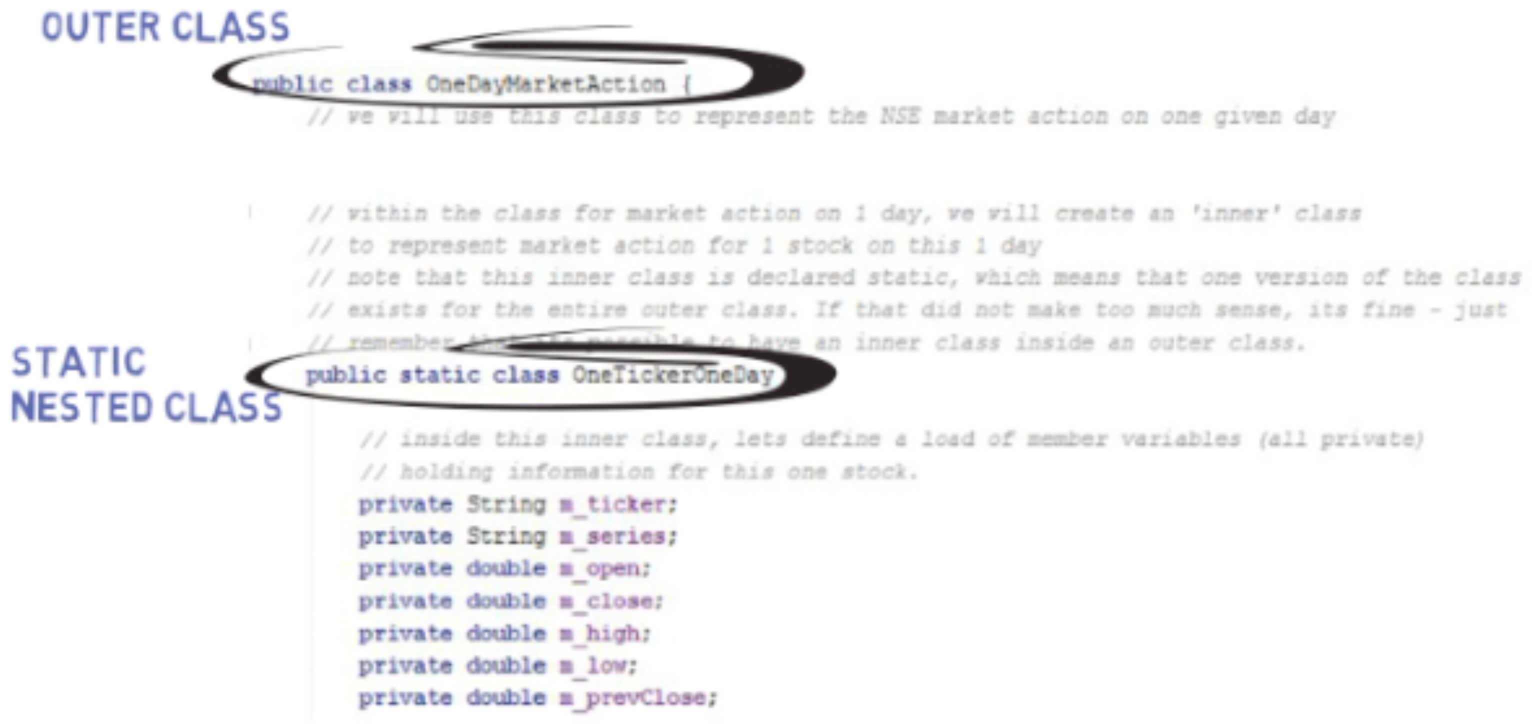
STATIC NESTED CLASSES

NON-STATIC NESTED CLASSES

ANONYMOUS INNER CLASSES

LOCAL CLASSES

STATIC NESTED CLASSES



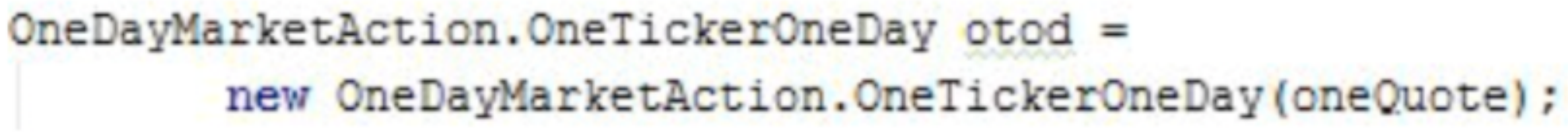
```
public class OneDayMarketAction {
    // we will use this class to represent the NSE market action on one given day


    // within the class for market action on 1 day, we will create an 'inner' class
    // to represent market action for 1 stock on this 1 day
    // note that this inner class is declared static, which means that one version of the class
    // exists for the entire outer class. If that did not make too much sense, its fine - just
    // remember that its possible to have an inner class inside an outer class.
    public static class OneTickerOneDay

        // inside this inner class, lets define a load of member variables (all private)
        // holding information for this one stock.
        private String m_ticker;
        private String m_series;
        private double m_open;
        private double m_close;
        private double m_high;
        private double m_low;
        private double m_prevClose;
```

STATIC NESTED CLASSES ARE USED WHEN THE INNER CLASS LOGICALLY MAKES SENSE INSIDE AN OUTER CLASS

NOTE THAT THIS DOES NOT MEAN THAT EVERY OBJECT OF THE NESTED CLASS EXISTS INSIDE AN OBJECT OF THE OUTER CLASS

IN FACT, OBJECTS OF THE NESTED CLASS ARE NOT ASSOCIATED WITH ANY SPECIFIC OBJECT OF THE OUTER CLASS - AND SO CAN NOT ACCESS PRIVATE MEMBERS OF THE OUTER CLASS

BUT THE INSTANTIATION INVOKES THE CONSTRUCTOR OF THE STATIC NESTED CLASS AS IF IT (THE CONSTRUCTOR) WERE A STATIC METHOD OF THE OUTER CLASS



```
OneDayMarketAction.OneTickerOneDay otod =
        new OneDayMarketAction.OneTickerOneDay(oneQuote);
```

OBJECTS OF THE STATIC NESTED CLASS CAN BE INSTANTIATED INDEPENDENTLY OF THE OBJECTS OF THE OUTER CLASS

THIS IS WHY OBJECTS OF THE NESTED CLASS ARE REFERRED TO AS STATIC.

NON-STATIC NESTED CLASSES, (AKA INNER CLASSES) ARE VERY DIFFERENT

```
class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

EACH OBJECT OF THE THE INNER CLASS ONLY EXISTS INSIDE A SPECIFIC OBJECT OF THE OUTER CLASS

```
OuterClass.InnerClass innerObject = outerObject.new InnerClass();
```

THUS AN OBJECT OF THE INNER CLASS MUST BE INSTANTIATED "INSIDE" AN OBJECT OF THE OUTER CLASS

BECAUSE ANY OBJECT OF THE INNER CLASS IS ASSOCIATED WITH A SPECIFIC OBJECT OF THE OUTER CLASS, INNER CLASSES CAN NOT HAVE ANY STATIC MEMBER VARIABLES

NON-STATIC NESTED CLASSES ARE A LOT LESS COMMONLY USED THAN STATIC NESTED CLASSES, BUT THEY HAVE 2 SPECIFIC USES THAT ARE WORTH MENTIONING

ANONYMOUS CLASSES (THINK OF THESE AS "USE-AND-THROW" CLASSES FOR ONE-TIME USE)

LOCAL CLASSES, WHICH CAN EXIST ANYWHERE IN A LOCAL SCOPE (FOR WHEN A CLASS WILL ONLY BE USED VERY LOCALLY, AND IT DOES NOT MAKE SENSE TO ADD TO THE CLASS HIERARCHY)