Using Custom Settings

It's possible to write programs that use extra customized settings from standard QuickFIX/J settings file. For example, let's say you want to limit the trade amount (price times quantity) for individual trades. You could place this parameter in the QuickFIX/J settings file.

```
[Session]
...
MaxOrderAmount=10000
...
```

Your application class could be defined with a constructor taking the settings file and calculate if the session constraint was violated.

```
public class MyApplication extends MessageCracker {
    private final SessionSettings settings;
    private final static String MAX_ORDER_AMOUNT_KEY = "MaxOrderAmount";

    public MyApplication(SessionSettings settings) {
        this.settings = settings;
    }

    public void onMessage(NewOrderSingle order, SessionID sessionID) {
        try {
            if (orderAmount(order) > maxTradeAmount(sessionID)) {
                sendBusinessReject("Trade amount too large.");
            } else {
                acceptOrder(order);
            }
        } catch (Throwable e) {
            handleException(sessionID, e);
        }
    }

    private double maxTradeAmount(SessionID sessionID) throws ConfigError,
  FieldConvertError {
        // This could be cached for performance, if needed
        return settings.getDouble(sessionID, MAX_ORDER_AMOUNT_KEY);
    }

    // ... other methods
}
```

You can also use the `SessionSettings.getSessionProperties` method to get a `Properties` object containing the settings (with defaults resolved) for a specified `SessionID`.

If you are writing your own QuickFIX/J main function you can use similar techniques to define your initiator or acceptor application, message store factory, and log factory classes. Put the class name in the settings file and use Java reflection to create the appropriate class for your application.

If you want different factories for each session (e.g., a file log for one session and a JDBC log for another) then you'll need to modify the acceptor or initiator implementation to support this functionality (a topic of a future tip).