

Event Based Programming

S. Hansen
UW - Parkside

T.V. Fossum
SUNY - Potsdam

Kenosha WI, May 23, 2010

Contents

1	Event Based Systems	1
1.1	Events	1
1.1.1	Responding to events	1
1.1.2	Event sources	2
1.2	Event based systems	2
1.2.1	Request-response	2
1.2.2	Message passing	2
1.2.3	Publish-subscribe	3
1.2.4	Events in Systems	3
1.2.5	System state	3
1.2.6	Event based systems defined	4
1.2.7	Discrete systems and events	4
1.2.8	Examples of events	5
1.3	Attributes of Event Based Systems	6
1.3.1	State Based	7
1.3.2	Nondeterminism	7
1.3.3	Loose Coupling	8
1.3.4	Decentralized Control	8
1.4	The Event Based Programming Paradigm	8
1.4.1	The Event Model	8
1.4.2	Events versus Method Invocations	9
1.4.3	Writing Event Based Programs	10
1.5	Goals of the Text	10
2	Event Based Programming Basics	13
2.1	Introduction	13
2.1.1	Integrated Development Environments	13
2.2	The Object–Oriented Event Model	14
2.2.1	A Simple Example	14
2.3	Java Language Features to Support Event Based Programming	17
2.3.1	Anonymous Classes	17
2.3.2	Inner Classes	19
2.3.3	List Example using Inner Classes	19
2.4	Inheritance	23
2.4.1	Model–View–Controller	23
2.4.2	The Drawing Application	33
2.5	List of Doubles	36
2.5.1	Double List Model	38

2.5.2	Average and Max Classes	40
2.5.3	The Main Class	42
2.5.4	Event Handler Classes	45
2.5.5	Event Propagation	48
2.6	Procedural Event Programming	48
2.7	Summary	49
3	Software Engineering Event Based Systems	51
3.1	Introduction	51
3.2	The Two Button Stopwatch	52
3.3	Event Based Software Development	53
3.4	Analyzing System Requirements	54
3.5	Design	57
3.5.1	Model Design	57
3.5.2	GUI Design	58
3.5.3	Control Design	58
3.6	Stopwatch Implementation	64
3.6.1	The Java Source Code	64
3.7	Testing	65
3.7.1	Testing Event Handling Code	66
3.7.2	Testing Event Sources	67
3.8	Debugging	67
3.8.1	Handler Registration	67
3.8.2	Nondeterminism	68
3.8.3	Cascading Events	68
3.9	Refactoring the Stopwatch	68
3.9.1	Control and View Control	68
3.9.2	The Modified Java Source Code	69
3.10	Summary	71
4	Event Infrastructure	73
4.1	Introduction	73
4.2	The Event Life Cycle	74
4.2.1	Event Handling Requirements	75
4.2.2	Timing Requirements	75
4.2.3	Concurrent Event Handling	75
4.3	Event Infrastructure Services	76
4.3.1	Event Registration	76
4.3.2	Event Dispatching	76
4.3.3	Hybrid Dispatching	80
4.4	Java's Support for Events	80
4.4.1	Handler Registration	80
4.4.2	JCL's Event Classes	81
4.4.3	Java Event Processing	81
4.5	Programmer Defined AWT Events	85
4.5.1	New Event Classes and Interfaces	87
4.5.2	The <code>BloodPressureListener</code> Interface	87
4.5.3	The <code>BloodPressureEvent</code> class	88
4.5.4	The <code>Patient</code> class	90
4.6	The Beans-Like Implementation	91

4.6.1	Revamped Blood Pressure Events	91
4.6.2	The Revamped Patient Class	92
4.6.3	Infrastructure or Client Code	93
4.7	Summary	93
5	Threads and Events	95
5.1	Introduction	95
5.2	Background	95
5.2.1	Threads and Concurrency	96
5.3	Why use Threads	96
5.4	A Multi-Threaded Event Handler	99
5.4.1	Simple Handler	100
5.4.2	Threaded Handler	100
5.5	Problems Arising from Multi-Threading	101
5.5.1	Problems with Resource Sharing	101
5.6	Threads as an Event Based System	102
5.6.1	State Based	103
5.6.2	Nondeterminism	104
5.6.3	Loose Coupling	104
5.6.4	Decentralized Control	105
5.7	Summary	105
6	Distributed Event Programming	107
6.1	Introduction	107
6.2	The Big Picture	107
6.3	Distributed System Infrastructures	108
6.4	Event Based and Distributed Systems	110
6.5	Publish - Subscribe	111
6.6	Challenges Developing Distributed Systems	111
6.7	The CORBA Infrastructure	112
6.7.1	Services, Tools and Classes	112
6.7.2	CORBA Programming Steps	115
6.8	Calculator Example	115
6.8.1	Defining the IDL	116
6.8.2	Compile the IDL file	116
6.8.3	Completing the Server	118
6.8.4	The Client	121
6.8.5	Starting the Application	124
6.9	Asynchronous Method Invocation	124
6.9.1	CORBA and Asynchronous Messages	124
6.9.2	Heart Monitor Example	125
6.10	Peer to Peer Distributed Applications	126
6.10.1	CORBA Peer to Peer Applications	126
6.10.2	Chat Example	126
6.11	Conclusion	130

7	Events and the Web	131
7.1	Introduction	131
7.1.1	Historical Perspective	132
7.1.2	Multi-tiered Architectures	132
7.2	Web Fundamentals	132
7.2.1	HyperText Transfer Protocol (HTTP)	133
7.2.2	HyperText Markup Language (HTML)	133
7.2.3	Web and Application Servers	135
7.3	Java Servlets	135
7.3.1	Calculator Example	135
7.4	Adding State to Web Applications	141
7.4.1	Cookies	141
7.4.2	Sessions	141
7.4.3	Other Ways to Maintain State	145
7.4.4	Combining Request and Response Pages	145
7.5	Java Server Pages (JSPs)	147
7.6	Web Services	149
7.6.1	Stock Quote Example	150
7.6.2	The eXtensible Markup Language (XML)	150
7.6.3	Finding a Web Service and its API	151
7.6.4	Summary of XML Uses in Web Services	152
7.7	Developing a Web Service	153
7.7.1	Quadratic Equations Revisited	153
7.7.2	Initial Steps	154
7.7.3	Web Service Annotations	154
7.7.4	Completing the Web Service	155
7.7.5	Testing and Deploying the Web Service	156
7.7.6	WSDL and Schema Revisited	156
7.8	Developing a Web Service Client	158
7.8.1	Initial Steps	158
7.8.2	Completing the Client	158
7.9	Making Web Services More Event Based	159
7.9.1	Developing a Web Service with State	160
7.9.2	Client for a Web Service with State	163
7.10	The Changing Landscape of Web Services	164
7.10.1	Web Services Inspection Language(WSIL)	164
7.10.2	SOAP versus JSON	164
7.10.3	RESTful Web Services	165
7.10.4	Evolving Language Tools	165
7.11	Conclusion	166

Preface

Introduction

This is a book about event based programming. There are dozens of computer science books that have 'event' or 'event based' in their titles. Almost all of these are about some particular language or system that uses events. This book is different. Our goal is to introduce you to event based programming – and, more generally, event based systems – as a computer science paradigm that focuses on the fundamental ideas relating to understanding, designing, implementing, and testing loosely coupled systems.

One of our objectives is to introduce you to approaches used in event based programming and to illustrate them with carefully crafted examples. Programming event based systems is different from procedural programming or object-oriented programming. If the solution is to be event based, programmers think about the problem differently. Its important that you understand why an event based approach is more appropriate for some problems than for others.

Event based programming also has its own set of challenges, principally because almost all event based systems are nondeterministic and are inherently difficult to test and debug. Another of our objectives is to show you how to use conceptual and design tools to create event based systems that behave correctly.

What you will learn from this book is applicable to any event based language or library. Since this is a book about programming, you will encounter many programs as you read it. We have chosen Java as our principal implementation language. Java is available almost universally for hardware platforms and operating systems, and Java has native support for events. However, this book is not about making you an expert in Java. Instead, you will gain an understanding of how to develop event based software, with Java serving as a particular implementation language. In the end, you will be able to develop reasonably sized event based applications in Java, and you will be able to take the principles and ideas you learn and apply them to new and different event based languages as needed.

Computing is continually changing. Computing professionals are always playing catch-up, trying to keep their technical knowledge current with regard to new languages, operating systems, and application versions. On the other hand, programming paradigms change very little. Procedural programming – with its while loops, if statements, and procedure calls – has remained largely unchanged since the 1950s. Similarly, the concepts behind object-oriented programming have been relatively stable for over 20 years. Procedural programming and object-oriented programming are *paradigms*. They aren't about any particular language. Instead, they give the programmer problem solving techniques and methodologies that are applicable in a variety of languages.

In recent years, event based programming has emerged as its own distinct paradigm. Its notions of runtime association of sources and handlers, and minimal timing assumptions apply, and will continue to apply, no matter how many new event based languages are released. The fundamental ideas behind the paradigm are stable. By emphasizing an understanding of the paradigm, we will

lay a foundation that will let you easily pick up new event based languages as they emerge.

Events are incredibly important in modern computing. They occur in graphical user interfaces (GUIs), operating systems, discrete event simulation, database management systems and many other computing-related fields. They are also integral to the operation of user interfaces in modern electronic devices such as cellular phones and television sets. Because events are so ubiquitous, it is reasonable to expect that you, as a student of computing, should understand them in detail. Unfortunately, you are more likely to learn first about event based programming when you are asked to develop GUIs – and then you would typically be exposed only to what you need to make the GUI operate. As you will see, events have many more applications than implementing GUIs.

Event based systems are commonplace, but they have distinct properties and pose unique challenges for developers. Event based programming deserves a broad-based comprehensive treatment in the computer science curriculum. This book is an effort to provide that treatment.

Audience

You should be comfortable approaching the material in this book if you have a background equivalent to a two-semester undergraduate-level course in programming with some experience using an object-oriented language. Knowing Java is a plus, but you should be able to understand the programming examples without it.

You should be comfortable dealing with elementary data structures such as arrays and lists. We may refer occasionally to more advanced data structures, in which case we will provide background discussion and pointers to resources where you can study relevant material. You should also be acquainted with classes, objects, inheritance, and polymorphism, though again we may take the opportunity to refresh your knowledge of these ideas when they play a particularly important role in our discussions or in example code. We do not expect you to have studied mathematics beyond elementary calculus and/or discrete math.

Resources

While Java is our primary implementation language in this book, other languages (C++, Python, and C#, for example) – along with appropriate library support – can serve equally well. We will post sample code on our course website (<http://cs.uwp.edu/Events>) to supplement the code in this book as it becomes available.