# EventHandling

Um eine Parallelisierung in unserem Programm zu ermöglichen, ist es notwendig eine asynchrone Kommunikation innerhalb zu ermöglichen. Diese Aufgabe erfüllen Events und die zugehörigen Verwaltungsklassen.

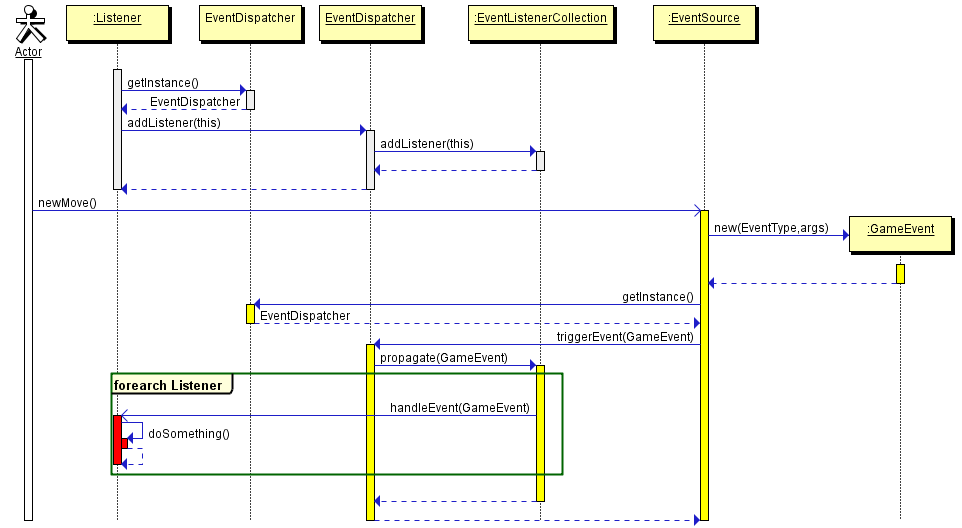
## Bestandteile

Folgende Bestandteile sind für das Event Handling implementiert worden.

1. Listener – Event Empfänger
2. Event Dispatcher – Zentrale Verwaltung
3. Event Listener Collection – Verwaltung der Empfänger
4. Event Source – Erzeuger eines Events
5. Event – Das Event(die asynchrone Nachricht)

## Ablauf

Im Folgenden soll das Zusammenspiel dieser Komponenten erläutert werden. In dem abgebildeten Sequenzdiagramm ist das genaue Vorgehen zu ersehen.



Die grauen Aktivitätsschritte beschreiben die Vorbereitungen zum Empfang eines Events. Der Listener, also derjenige, der ein Event empfangen möchte, muss sich als Listener zentral registrieren. Dazu kontaktiert er den EventDispatcher. Dieser Singleton übernimmt die zentrale Verwaltung für die Events. Über die Methode addListener() kann sich ein Listener registrieren. Im EventDispatcher wird dieser Listener dann in eine EVentListenerCollection aufgenommen. Diese Collection wird für jeden Event Typ erzeugt. Da wir nur ein Event Typ, nämlich das Game Event mit diversen Eigenschaften benötigen, wird nur eine Collection erstellt.

Mit den gelben Aktivitäten wird nun das Auslösen eines Events beschrieben. Aus der EventSource, dem Sender, wird eine neue Instanz von GameEvent erzeugt. Nun muss diese Event asynchron versendet werden. Dazu wird der Dispatcher über triggerEvent() benachrichtigt. Dieser ruft nun die Collection mit der methode propagate(). In dieser Methode wird nun für jeden Listener die Methode handleEvent() aufgerufen. Diese Methode ist als Interface definiert und jeder Listener muss diese Methode implementieren.

Bis jetzt ist dieser Aufruf jedoch synchron. Die roten Aktivitäten zeigen nun einen asynchronen Aufruf. Für jeden Listener wird die handleEvent() Methode in einem neuen Thread gestartet und läuft somit unabhängig von der EventListenerCollection weiter.

Mit dieser Architektur kann nun parallel im Programm gearbeitet werden und die heute verwendeten Multicoresysteme deutlich besser ausgenutzt werden.