Custom events in Java

1. Very simple example of custom events (callback method with no parameters):

**import** java.util.ArrayList;

// listener interface

**interface** IListener {

**public** **void** action(); // callback method with no parameters

}

// class that can be observed

**class** Human {

ArrayList<IListener> listenersList = **new** ArrayList<>();

// add listener to list of listeners

**public** **void** addListener(IListener aListener) {

listenersList.add(aListener);

}

// inform all listeners

**private** **void** notifyListeners() {

**for**(IListener e : listenersList)

e.action();

}

// action that requires informing the listeners

**public** **void** displayFood() {

System.*out*.println("Human: I am displaying some food");

notifyListeners(); // the human notifies those who are listening

}

}

// a listener class

**class** Dog **implements** IListener {

@Override

**public** **void** action() {

System.*out*.println("Doggo: bork bork");

}

}

// main class

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Human kim = **new** Human(); // Kim is the human

Dog rex = **new** Dog(); // Rex is the dog

kim.addListener(rex); // add Rex to Kim’s list of listeners

kim.displayFood(); // Kim displays food

}

}

2. What will happen if Rex listens to a second person? How to distinguish between these people? What will happen if we want the dog to be able to listen to events other than food being shown? To model this, we can add a parameter to the callback method: an event.

**import** java.util.ArrayList;

// the event class that will give listeners additional information about events and their sources

**class** HumanEvent {

Human source;

String description;

**public** HumanEvent(Human source, String description) {

**this**.source = source;

**this**.description = description;

}

}

// listener interface

**interface** IListener {

**public** **void** action(HumanEvent evt); // callback method with event parameter

}

// class that can be observed

**class** Human {

String name;

ArrayList<IListener> listenersList = **new** ArrayList<>();

**public** Human(String name) {

**this**.name = name;

}

// add listener to list of listeners

**public** **void** addListener(IListener aListener) {

listenersList.add(aListener);

}

// inform all listeners

**private** **void** notifyListeners(String desc) {

HumanEvent evt = **new** HumanEvent(**this**, desc);

**for**(IListener e: listenersList)

e.action(evt);

}

// actions that requires informing the listeners

**public** **void** displayFood() {

System.*out*.println("Human " + name + ": I am displaying food");

notifyListeners("displaying food");

}

**public** **void** standUp() {

System.*out*.println("Human " + name + ": I am standing up");

notifyListeners("standing up");

}

}

// a listener class

**class** Dog **implements** IListener {

String name;

**public** Dog(String name) {

**this**.name = name;

}

@Override

**public** **void** action(HumanEvent evt) {

System.*out*.println("Doggo " + name +

": bork, my human " + evt.source.name + " is " + evt.description);

}

}

// main class

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Human kim = **new** Human("Kim"); // Kim is a human

Human peabody = **new** Human("Mr. Peabody"); // Mr. Peabody is another human

Dog rex = **new** Dog("Rex"); // Rex is a dog

Dog moxie = **new** Dog("Moxie"); // Moxie is another dog

kim.addListener(rex); // add Rex to Kim’s list of listeners

kim.addListener(moxie); // do the same for Moxie

peabody.addListener(rex); // add Rex to Mr. Peabody’s list of listeners

peabody.addListener(moxie); // do the same for Moxie

kim.displayFood(); // Kim displays some food

kim.standUp(); // Kim stands up

peabody.standUp(); // Mr. Peabody stands up

}

}