Assumptions made:

- One co-ordinate/location can only have 1 event.
- An event has 0 or more tickets and all tickets are priced the same for that event.
- There is a random amount of events in the grid. This is generated upon execution of the app.

How might I change my program to support multiple events at the same location?

- I would make use of the linked list data structure for this. The Event object would act as a linked list by referring to other Events in the same location. This will be ideal as the linked list is far more efficient in terms of adding and deleting within the linkedlist compared to arrays. For example:
  Event A is in location (4,4). Event B and C is also in location (4,4). Thus, Event A would point to event B and Event B would point to Event C.
- The downside of using linkedlist is that it will use more memory compared to arrays although this is a good trade off as with cloud computing is in the rise giving customers more memory to work with.
- To improve this method, you could even use a doubly linkedlist so you'd be able to traverse the list of events quicker because it would allow you to search forwards and backwards.

How would you change your program if you were working with a much larger world size?

- I would use data structures that would be the most efficient given the situation. For this I would use a red-black tree structure as it has a general time complexity of log(n) therefore searching an event using this structure would be quick compared to other structures.

  E.g. if the tree was structured with the root node resembling an event at 0,0 and the bottom of the leafs at the bottom of the tree resembled the events at the bottom right of the grid. Then whatever event the user wanted to know which events were closeby, we'd use the user input coordinates to traverse through the tree and land on a node in which it will give us the event either on that location or the closest event to that location. From there we'd get a cluster of events around the node we landed on and then calculate the Manhattan distance on the events we have in our cluster. This approach would be far more efficient than brute forcing the calculation on every event there is on the grid.

- I would also use threads on my application and make use of the structures that are catered towards concurrency such as the ConcurrentHashMap found in the concurrency collections package in Java.