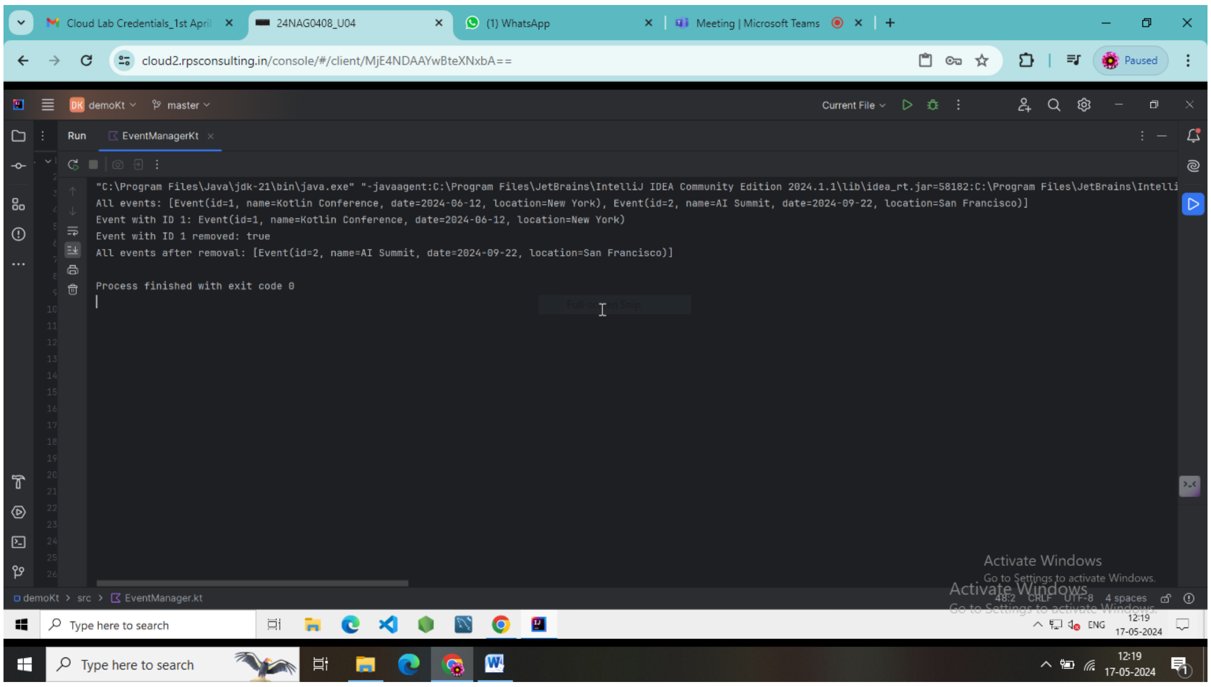
# Day-2, Assignment-2

**Task 5:** Design a EventManager class with methods to add and remove events.

```
data class Event(
   val id: Int,
   val name: String,
   val date: String,
   val location: String
)
class EventManager {
   private val events = mutableMapOf<Int, Event>()
   private var nextId = 1
   fun addEvent(name: String, date: String, location: String): Int {
      val event = Event(nextId, name, date, location)
      events[nextId] = event
      return nextId++
   }
   fun removeEvent(eventId: Int): Boolean {
      return events.remove(eventId) != null
   }
   fun getEvent(eventId: Int): Event? {
      return events[eventId]
   }
   fun listEvents(): List<Event> {
      return events.values.toList()
   }
}
```

```
fun main() {
    val manager = EventManager()
    // Add events
    val event1Id = manager.addEvent("Kotlin Conference", "2024-06-12", "New York")
    val event2Id = manager.addEvent("AI Summit", "2024-09-22", "San Francisco")
    // List events
    println("All events: ${manager.listEvents()}")
    // Get an event
    println("Event with ID $event1Id: ${manager.getEvent(event1Id)}")
    // Remove an event
    val removed = manager.removeEvent(event1Id)
    println("Event with ID $event1Id removed: $removed")
    // List events after removal
    println("All events after removal: ${manager.listEvents()}")
}
```

## Output:

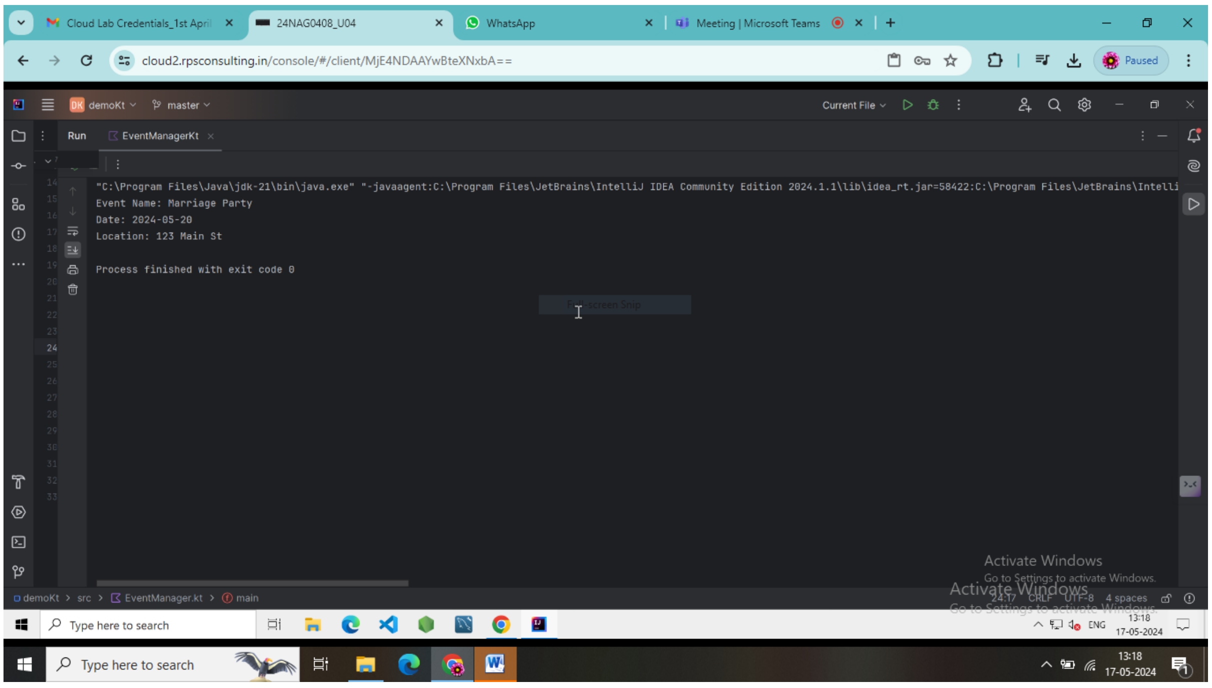


```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.1\lib\idea_rt.jar=58182:C:\Program Files\JetBrains\Intelli
All events: [Event(id=1, name=Kotlin Conference, date=2024-06-12, location=New York), Event(id=2, name=AI Summit, date=2024-09-22, location=San Francisco)]
Event with ID 1: Event(id=1, name=Kotlin Conference, date=2024-06-12, location=New York)
Event with ID 1 removed: true
All events after removal: [Event(id=2, name=AI Summit, date=2024-09-22, location=San Francisco)]

Process finished with exit code 0
```

**Task 6:** Create a Display interface with a method to show event details and implement it in the EventManager.

```kotlin
// Define the Display interface

interface Display {

    fun showEventDetails(eventName: String, eventDetails: String)

}

// Implement the Display interface in the EventManager class

class EventManager : Display {

    override fun showEventDetails(eventName: String, eventDetails: String) {

        println("Event Name: $eventName")

        println("Event Details: $eventDetails")

    }

}

// Main function to demonstrate the functionality

fun main() {

    val eventManager = EventManager()

    eventManager.showEventDetails("Kotlin Conference", "A conference about Kotlin programming
language.")

}
```

## Output:

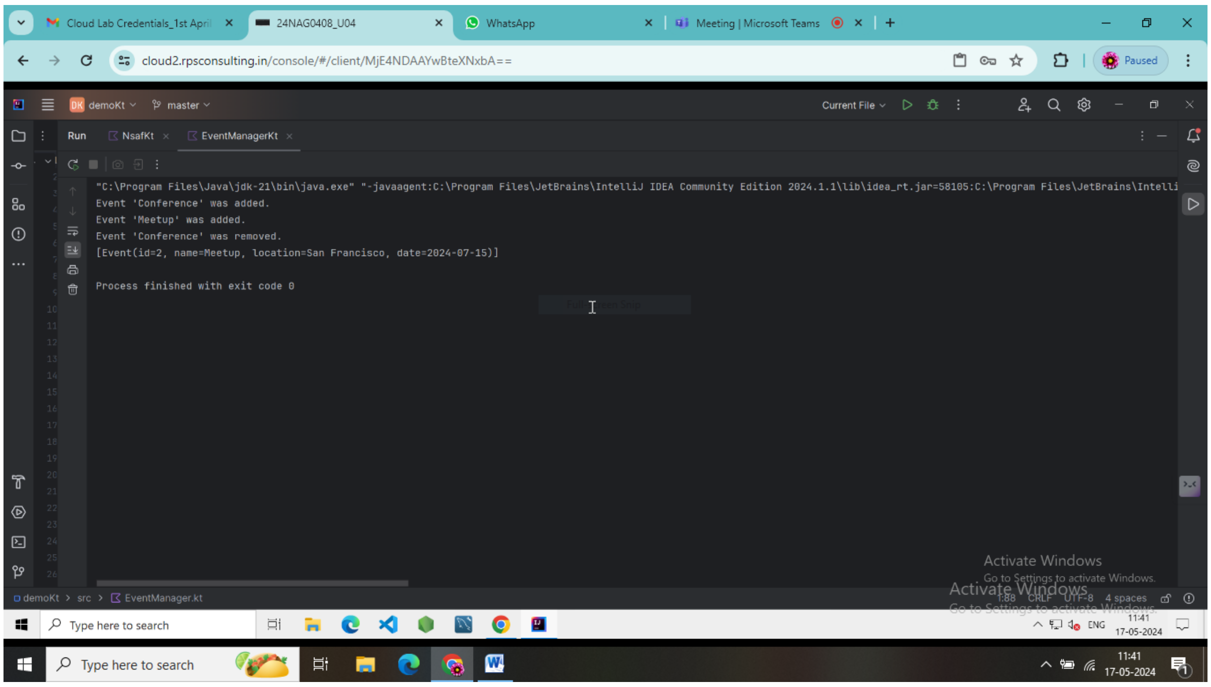


```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.1\lib\idea_rt.jar=58422:C:\Program Files\JetBrains\Intelli
Event Name: Marriage Party
Date: 2024-05-20
Location: 123 Main St

Process finished with exit code 0
```

**Task 7:** Utilize higher-order functions to implement a simple notification system for event updates.

```
data class Event(val id: Int, val name: String, val location: String, val date: String)

fun main() {

  val eventManager = EventManager()

  // Register a listener for event updates

  eventManager.registerListener { event, action ->

    println("Event '${event.name}' was $action.")

  }

  val event1 = Event(1, "Conference", "New York", "2024-06-01")

  val event2 = Event(2, "Meetup", "San Francisco", "2024-07-15")

  // Add events

  eventManager.addEvent(event1)

  eventManager.addEvent(event2)

  // Remove an event

  eventManager.removeEvent(1)

  // Print all events

  println(eventManager.getAllEvents())

}

class EventManager {

  private val events = mutableListOf<Event>()

  private val listeners = mutableListOf<(Event, String) -> Unit>()

  fun addEvent(event: Event): Boolean {

    return if (events.none { it.id == event.id }) {

      events.add(event)

      notifyListeners(event, "added")

      true

    } else {
            false // Event with the same id already exists
        }
    }
    fun removeEvent(eventId: Int): Boolean {
        val event = events.find { it.id == eventId }
        return if (event != null) {
            events.remove(event)
            notifyListeners(event, "removed")
            true
        } else {
            false
        }
    }
    fun getAllEvents(): List<Event> {
        return events.toList()
    }
    fun findEventById(eventId: Int): Event? {
        return events.find { it.id == eventId }
    }
    fun registerListener(listener: (Event, String) -> Unit) {
        listeners.add(listener)
    }
    private fun notifyListeners(event: Event, action: String) {
        listeners.forEach { it(event, action) }
    }
}
```

## Output:



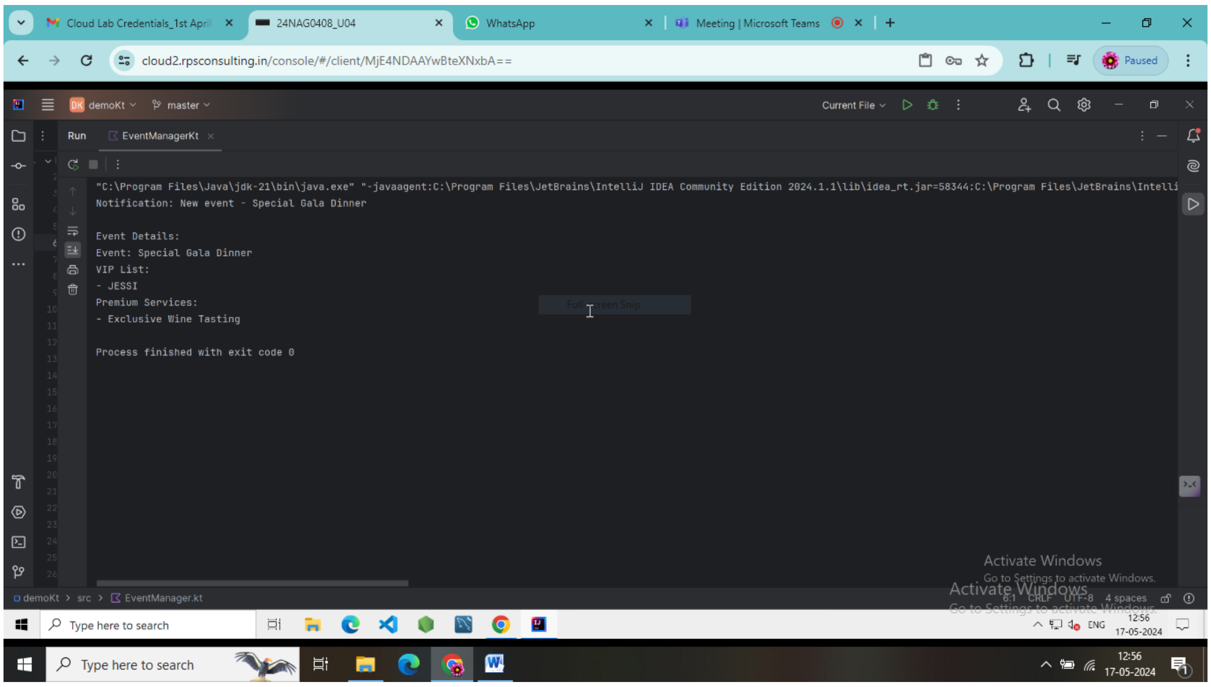
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.1\lib\idea_rt.jar=58105:C:\Program Files\JetBrains\Intelli
Event 'Conference' was added.
Event 'Meetup' was added.
Event 'Conference' was removed.
[Event(id=2, name=Meetup, location=San Francisco, date=2024-07-15)]

Process finished with exit code 0
```

**Task 8:** Construct subclass SpecialEvent with additional features like VIP lists and premium services.

```
import java.util.*

interface Display {

   fun showEventDetails(event: String)

}

open class EventManager : Display {

   protected val events: MutableSet<String> = TreeSet()

   protected val subscribers: MutableList<(String) -> Unit> = mutableListOf()

   fun addEvent(event: String) {

      events.add(event)

      notifySubscribers(event)

   }

   fun removeEvent(event: String) {

      events.remove(event)

      notifySubscribers("$event has been removed.")

   }
```

```
    fun subscribe(subscriber: (String) -> Unit) {
        subscribers.add(subscriber)
    }
    fun unsubscribe(subscriber: (String) -> Unit) {
        subscribers.remove(subscriber)
    }
    private fun notifySubscribers(event: String) {
        for (subscriber in subscribers) {
            subscriber(event)
        }
    }
    override fun showEventDetails(event: String) {
        println("Event: $event")
        // Add any additional details you want to display for the event
    }
    fun displayEvents() {
        if (events.isEmpty()) {
            println("No events scheduled.")
        } else {
            println("Events:")
            for ((index, event) in events.withIndex()) {
                println("${index + 1}. $event")
            }
        }
    }
}
class SpecialEvent : EventManager() {
    private val vipList: MutableSet<String> = TreeSet()
    private val premiumServices: MutableList<String> = mutableListOf()
    fun addToVIPList(person: String) {
        vipList.add(person)
    }
    fun removeFromVIPList(person: String) {
        vipList.remove(person)
    }
    fun addPremiumService(service: String) {
        premiumServices.add(service)
    }
    fun removePremiumService(service: String) {
        premiumServices.remove(service)
    }
    override fun showEventDetails(event: String) {
        super.showEventDetails(event)
        println("VIP List:")
        for (person in vipList) {
            println("- $person")
        }
        println("Premium Services:")
        for (service in premiumServices) {
            println("- $service")
        }
    }
}
fun main() {
    val specialEvent = SpecialEvent()

    // Subscribe to event updates

    specialEvent.subscribe { event ->

        println("Notification: New event - $event")

    }

    specialEvent.addEvent("Special Gala Dinner")

    specialEvent.addToVIPList("JESSI")

    specialEvent.addPremiumService("Exclusive Wine Tasting")

    println("\nEvent Details:")

    specialEvent.showEventDetails("Special Gala Dinner")

}
```

## Output:



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.1\lib\idea_rt.jar=58344:C:\Program Files\JetBrains\Intelli
Notification: New event - Special Gala Dinner

Event Details:
Event: Special Gala Dinner
VIP List:
- JESSI
Premium Services:
- Exclusive Wine Tasting

Process finished with exit code 0
```