

NATIONAL SCHOOL OF ARTS AND SCIENCES

COMP1005 – OBJECT ORIENTED PROGRAMMING SEMESTER 2 (S2)

UNIVERSITY OF NOTRE DAME , SYDNEY

A PROJECT REPORT ON

JAVA EVENTS MANAGEMENT SYSTEM (EMS)

SUBMITTED BY:

SAJINA MAGAR

STUDENT ID : 32090360

SUBMITTED TO : MR KASHIF NISAR

SCHOOL OF ARTS AND SCIENCES

3 OCTOBER , 2025

**INTRODUCTION**

In this modern world , events plays a vital role in academic , professionals as well as in social environments. Universities has their own seminars and workshops , companies holds conferences , product launches and so on.

The Event Management System (EMS) developed in this project is a simple,console based java application which particularly focuses on managing the events properly using the principles of Object-Oriented Programming (OOP).This project uses the OOP concepts like Encapsulation, Inheritance, Polymorphism, and Abstraction which can be combined to design modular, reusable, and maintainable software.

By using Java as a programming language , the EMS gets the strong type safety,extensive library support and platform independence. This system supports the modern world entities such as event , venue , participant , organizer as the classes where it contains the attributes and methods in it which defines their properties and behavior.

This project serves two main purposes which includes :

* To develop a functional event management system
* To practically demonstrate OOP design principles.

It acts as the bridge between theoretical learning and real world application by how the concepts can be directly applied to everyday situations.

**OBJECTIVES**

The primary objectives of this project is to design and implement the Simple Event Management System using Java .This system aims to provide a structured , modular solution for the management of events , participants , and venues while using the concepts such as classes, objects and relationships in software design process.

Specific objectives includes:

1. To create a object-oriented model of real world entities that includes events , venues , participants and organizers.
2. To apply the concepts such as Encapsulation , Inheritance , Polymorphism and abstraction.
3. To show method overloading , method overriding and proper use of interfaces using Java.
4. To enhance code reusability , readability and maintainability in the overall software design.

These are the objectives which tells about the project functionality as well as reinforces fundamental OOP concepts in practice.

**CLASS DESIGN**

This project Event Management System(EMS) follows a modular object oriented approach where java classes are used to represent the entities which consists of attributes ,methods and its purpose.

This section explains about the major classes , attributes , methods and its purpose.

1. **EVENT CLASS**

It represents the general blueprint for all types of events . It holds the common common behaviors as well as attributes shared by workshops , concerts and seminars.

Attributes are : eventName , eventDate , venue

Methods are : setVenue( Venue v ), displayDetails()

These includes the name of the event , scheduled date , associated venue object and methods to assign a value , abstract method overriden by subclasses.

1. **WORKSHOP / SEMINAR / CONCERT (SUBCLASSES OF EVENT)**

It extend the Event class to represent specific types of events . Each subclasses customizes behavior using the method overriding.

Attributes are : topic ,speaker ,performer

Methods are : displayDetails()

This demonstrates the inheritance and polymosphism as all are treated as Event objects but behaves differently in the runtime.

1. **VENUE CLASS**

This represents the physical location where the event is being hosted.

Attributes are : venueName , capacity , location

Methods are : checkAvailability() ,displayVenue()

Encapsulation is applied here which keeps the attributes private and provides getter and setter function.

1. **PARTICIPANT CLASS**

It represents a user registering for the particular events.

Attributes are : participantName , email

Methods are :

* public void registerEvent(Event e)
* public void displayParticipant()

1. **ORGANIZER CLASS**

It represents the person or the authority who is responsible for creating and managing the events.

Attributes are : organizerName , contactInfo

Methods are :

* public void createEvent()
* public void manageEvent()

1. **MAIN CLASS**

It acts as the driver class where objects are created , methods calling ,OOP concepts and principles are demonstrated.

Methods are :

* public static void main(String[ ] args)

**ENCAPSULATION IN THE EMS**

Encapsulation is one of the four fundamental principles of OOP. It is the practice of hiding the internal details and only exposing the necessary details using the public methods. This basically ensures integrity of data , security and provides the controlled access to class attributes.

class Participant {

private String participantName;

private String email;

public String getParticipantName() {

return participantName;

}

public void setParticipantName(String participantName) {

this.participantName = participantName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

By using the private fields , the EMS prevents the unauthorized or accidental modification or changes in the actual sensitive information.

Hence , encapsulation in OOP and in this EMS ensures that each entity’s data is protected and accessed only in a meaningful as well as in a controlled manner which must align with the principle of secure and reliable software design.

**POLYMORPHISM IN THE EMS**

Polymorphism is one of the main principle in the Object Oriented Programming which allows the same named method to perform in a different way depending on the object that calls it.

In the Event Management System (EMS) , polymorphism is implemented in two main ways which includes:

* Method overriding (runtime polymorphism)
* Method overloading (compile time polymorphism)

**METHOD OVERRIDING**

Run time polymorphism occurs when a subclass provides a specific implementation of a method which is already being defined in the parent class . In the EMS , the base class Event has the abstract method called displayDetails() which can be overriden in each subclasses.

abstract class Event {

abstract void displayDetails();

}

class Workshop extends Event {

void displayDetails() {

System.out.println("This is a Workshop event focusing on technical learning.");

}

}

class Seminar extends Event {

void displayDetails() {

System.out.println("This is a Seminar event featuring guest speakers.");

}

}

These above can be executed in the following manner :-

*Event* e1 = new Workshop();

*Event* e2 = new Seminar();

e1.displayDetails();

e2.displayDetails();

The system determines at runtime which version of **displayDetails()** to call which demonstrate the dynamic method dispatch.

**METHOD OVERLOADING**

Compile time polymorphism allows the multiple methods with the same name but with different parameter lists. In the EMS, this concept is demonstrated in the Participant class, where the participants can register for events in different ways according to the respective parameters:

class Participant {

void registerEvent(Event e) {

System.out.println("Participant registered for an event.");

}

void registerEvent(Event e, String feedback) {

System.out.println("Participant registered and gave feedback: " + feedback);

}

}

Polymorphism in EMS enhances as well as supports flexibility and code reusability.It simplifies the code which makes it more readable which eventually makes the design of the software more dynamic and maintainable.

**INHERITANCE IN THE EMS**

Inheritance is one of the fundamental concept in the Object Oriented Programming which usually focuses on acquiring the properties and behaviour of the existing classes ( parent ,superclass ) by a new sub class of a child class. It promotes code reusability , hierarchical classification and easier maintanance.

The base class Event defines the general structure for the events in the system and other subclasses like workshop , seminar and concert extends the Event class to reuse these attributes while adding their own specific characteristics.

class Event {

String eventName;

String eventDate;

void displayDetails() {

System.out.println("General Event Information");

}

}

class Workshop extends Event {

String topic;

@Override

void displayDetails() {

System.out.println("Workshop on: " + topic);

}

}

class Concert extends Event {

String performer;

@Override

void displayDetails() {

System.out.println("Concert performed by: " + performer);

}

}

If new types of events are to be added in the future they can simply extend the Event class for better and systematical solution.

# **ABSTRACTION AND INTERFACES IN THE EMS**

Abstraction is the process of hiding the unnecessary implementation and only showing the informations , and essential features that are relavant of an object. In this Event Management System , abstraction is achieved using abstract classes and interfaces.

Each subclasses such as Workshop , Seminar or Concert must provide its own version of the displayDetail() method to ensure the specific behavior is implemented while also maintaining the consistent interfaces accross all the event types.

abstract class Event {

abstract void displayDetails();

}

interface Schedulable {

void scheduleEvent();

void cancelEvent();

}

Using this ensures the abstraction making the EMS more flexible , modular and easy to extend.

**SCREENSHOTS AND DEMONSTRATION**

Figure 1: EMS Project Folder Structure

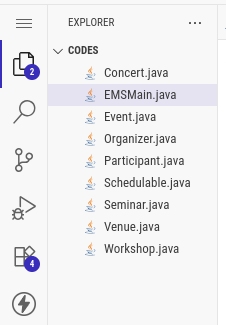


Fig. folder project structure.

Figure 2: Encapsulation in Participant.java



Fig. encapsulation in Java (EMS)

Figure 3: Inheritance and Polymorphism in Workshop.java

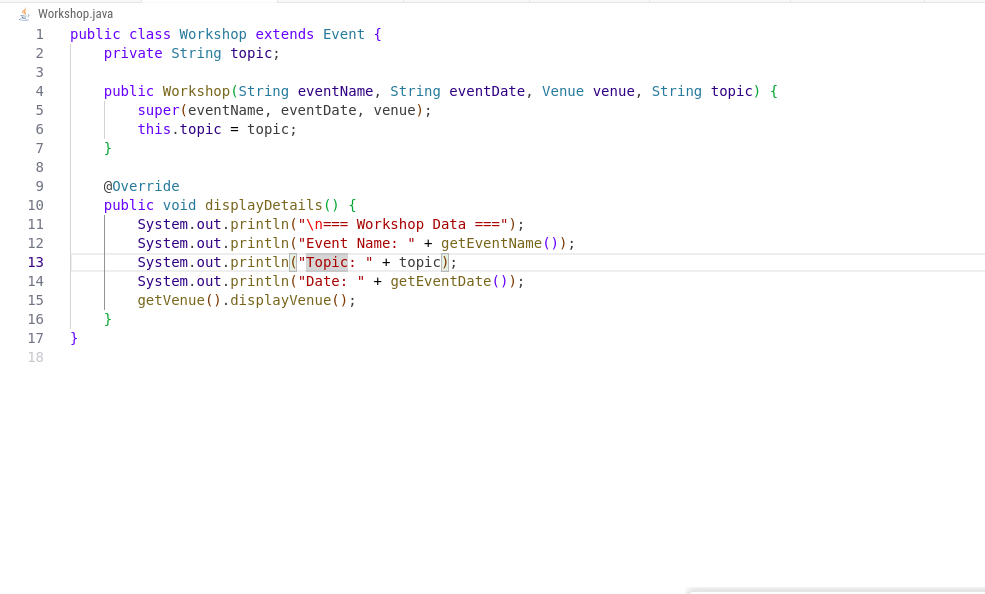


Fig. inheritance and polymorphism

Figure 4: Abstraction and Interface Implementation

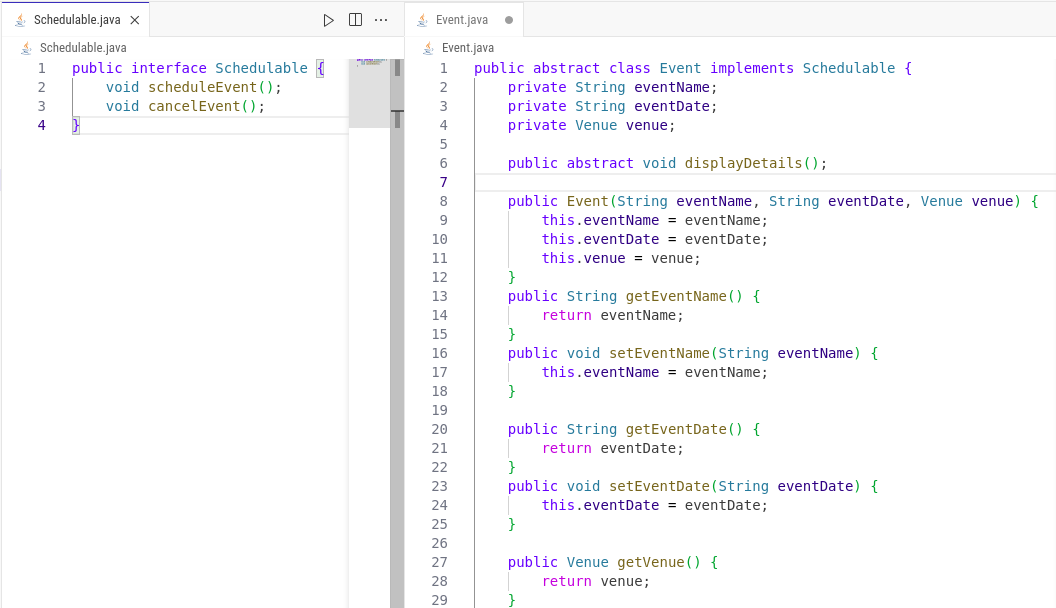


Fig. abstraction and interface

Figure 5: Compilation of EMS Source Code

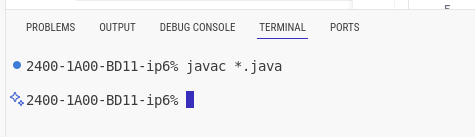


Figure 6: Execution Output of EMS Program

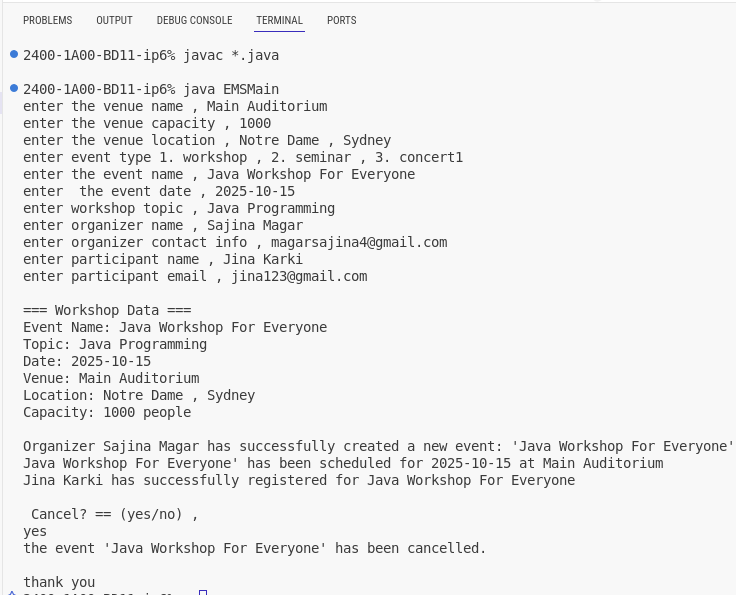


Fig. output of the program

**REFLECTION ON CHALLENGES AND LESSONS LEARNED**

Developing the Event Management System in java was indeed a valuable experience that boost up my confidence , my skills in programming and my understanding in Object Oriented Programming (OOP). Initially I faced challenges figuring out the java file structure , relationships between classes and compilation errors which were giving such a headache but things got right afterward after some research

Overall this project gave me a solid understanding on how the theoretical OOP concepts are used in real world system . It improved my logical thinking, debugging skills and ability to design the software using Java.

**CONCLUSION**

In conclusion this project is the practical use case of the theoretical things which i studied in the classroom. This project manages the Event in more better approach as compared to the traditional approach.