Table of Contents

1. Problem Statement

Design a Catering Management System which has the ability to manage catering services of a Catering Company

2. Software Requirements

According to detailed problem statement, this software should have ability to handle an event with attributes like, name of event, number of guests, customer name, total food cost, number of minutes spent on event, cost per minute, and average cost per person. However few other attributes (like total event cost) that were not specified in problem statement were also added to add extra functionalities in software.

3. Assumptions

It is assumed that there is no traveling cost for catering staff. Initially price of food is set to be 250 per person which is changeable during runtime of software.

4. Design Process (Modeling)

This application has been design by recognizing use case and actors of this application.

4.1. Actors

The actor of this software could be man in Catering Management Company that is responsible for recording all details of events

4.2. Use Case

Here are following use cases:

1. Start Program
2. Add a new event
3. Display all events
4. Change Settings
5. Close the Program

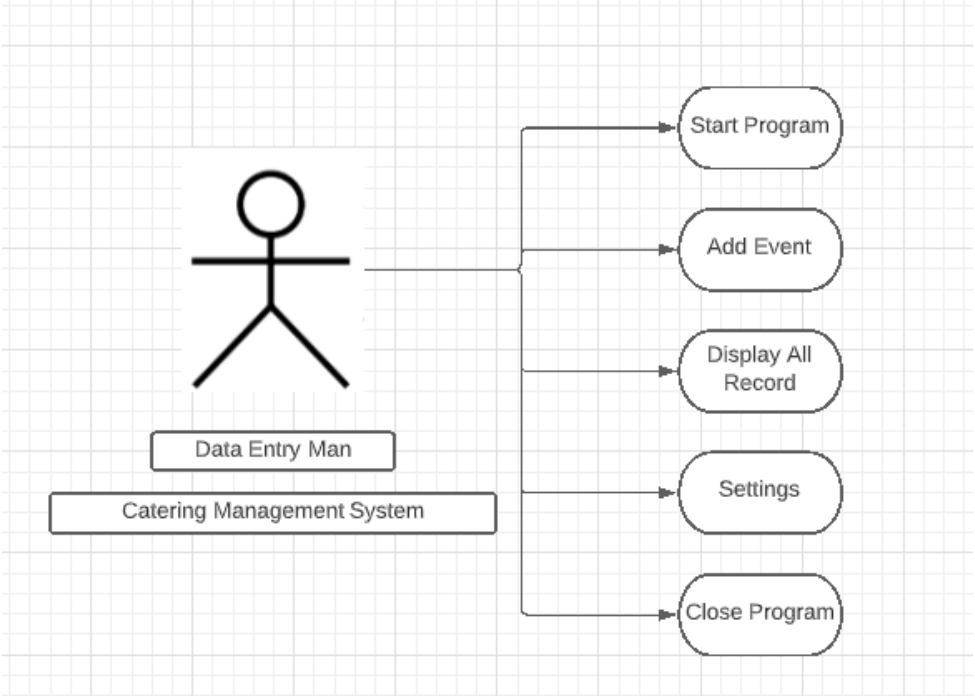


Figure 1: Use Case UML Diagram

4.3. Use Case to Classes

"Event" noun has been used as a very important thing in problem statement. So, this software has an abstract class **Event** from which four more classes are inherited. Here we have made a function called setCostPerMinute as pure virtual function.

The other classes which are inherited are:

A. Wedding

B. Birthday

C. Festival

D. General (can be used for general events)

As depicted below,

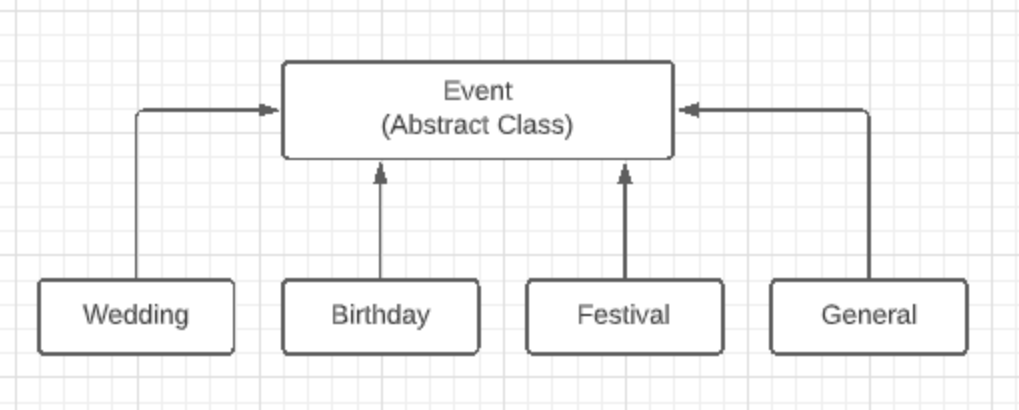


Figure 2: Inheritance Heirarchy of Software

5. Application Details

This application has five classes.

5.1. Event Class

It is main class that is abstract it has following data members.

```
string  nameOfEvent;
        customerName;
        locationOfEvent;
        occuringTimeOfEvent;

int  foodCostPerPerson  =  250;
     totalFoodCost;
     numberOfMinutesSpent;
     totalEventCost;
```

It has following behaviors,

```
Event();

void  setNameOfEvent(string);
      setCustomerName(string);
      setLocationOfEvent(string);
      setOccuringTimeOfEvent(string);
      setNumberOfGuests(int);
      setNumberOfMinutesSpent(int);
      setFoodCostPerPerson(int x);
          void  setCostPerMinute(int)  =  0;

string  getNameOfEvent();
        getCustomerName();
        getLocationOfEvent();
        getOccuringTimeOfEvent();
```

```
int getNumberOfGuests();
    getNumberOfMinutesSpent();
    getCostPerMinute();
    getTotalFoodCost();
    getFoodCostPerPerson();
    getTotalEventCost();
      getAverageCostPerPerson();

void calcTotalFoodCost();
     calcTotalEventCost();
     calcAverageCostPerPerson();

void diskIn();
     diskOut();

void changeFoodCostTo(int);
```

5.2. Wedding Class

It only overrides setCostPerMinute function,

```
void setCostPerMinute(int);
```

5.3. Birthday Class

It has only one function that overrides setCostPerMinute function

```
void setCostPerMinute(int);
```

5.4. Festival Class

It has only one function that overrides setCostPerMinute function

```
void setCostPerMinute(int);
```

5.5. General Class

It has only one function that overrides setCostPerMinute function

```
void setCostPerMinute(int);
```

5.6. Main.cpp

It has three functions to perform different functionalities

```
void add();
void read();
void settings();
```

5.6.1. void add()

It adds a new event.

```
system("CLS");
cout << setw(50) << "Catering Management System" << endl
cout << "Select Type of Event" << endl;
cout << "   1. Wedding" << endl;
cout << "   2. Birthday" << endl;
cout << "   3. Festival" << endl;
cout << "   4. General" << endl;
cout << "   5. Back" << endl;
cin >> ch;
switch (ch) { ... }
```

5.6.2. void read()

It allows to read the whole events record

5.6.3. void settings()

It allows to change food cost per person price (which was set 250 by default).

It also allow to change cost per minute for all four classes.

```
system("CLS");
cout << setw(50) << "Catering Management System" << endl;
cout << "Settings: " << endl;
cout << "1. Change Food Cost Per Person" << endl;
cout << "2. Change Cost Per Minute (for Weddings)" << endl;
cout << "3. Change Cost Per Minute (for Birthdays)" << endl;
cout << "4. Change Cost Per Minute (for Festivals)" << endl;
cout << "5. Change Cost Per Minute (for General Events)" << end
cout << "6. Back" << endl;
cin >> ch;
switch (ch)
{
case '1':
 { ... }
case '2':
 { ... }
case '3':
 { ... }
```

6. Data Management

This all data has been stored in a file through the concept of File Handling.

```
void  diskIn();
```

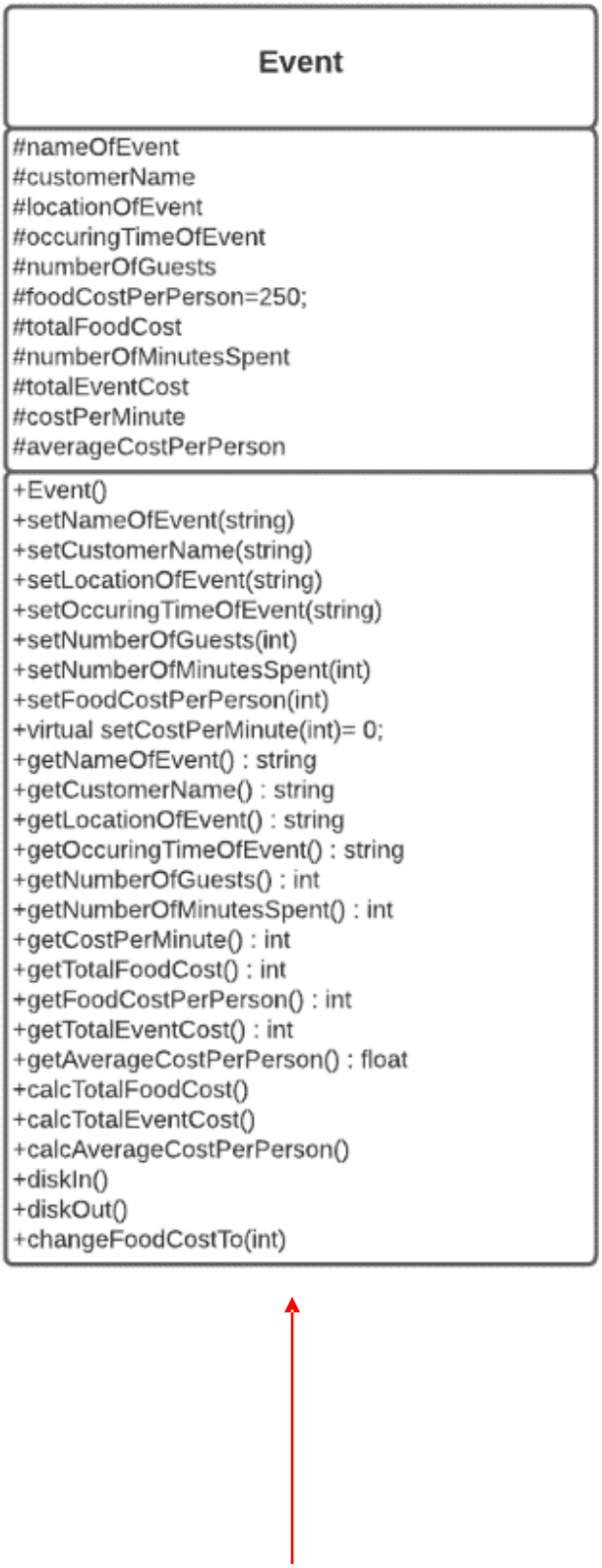
```
void Event::diskIn()
{
    ofstream write;
    write.open("E:\\Degree Courses\\Semester 3\\CS 212 - Object Oriented P
    write << "Name of Event: " << getNameOfEvent () << endl;
    write << "Name of Customer: " << getCustomerName () << endl;
    write << "Location of Event: " << getLocationOfEvent () << endl;
    write << "Occuring Time of Event: " << getOccuringTimeOfEvent () << en
    write << "Number of Guests: " << getNumberOfGuests () << endl;
    write << "Total Food Cost: " << getTotalFoodCost() << endl;
    write << "Number of Minutes Spent: " << getNumberOfMinutesSpent() << e
    write << "Average Cost Per Person: " << getAverageCostPerPerson() << e
    write << "Cost Per Minute: "
    write << "Total Event Cost: "
    write << "..................
    write.close();
}
```

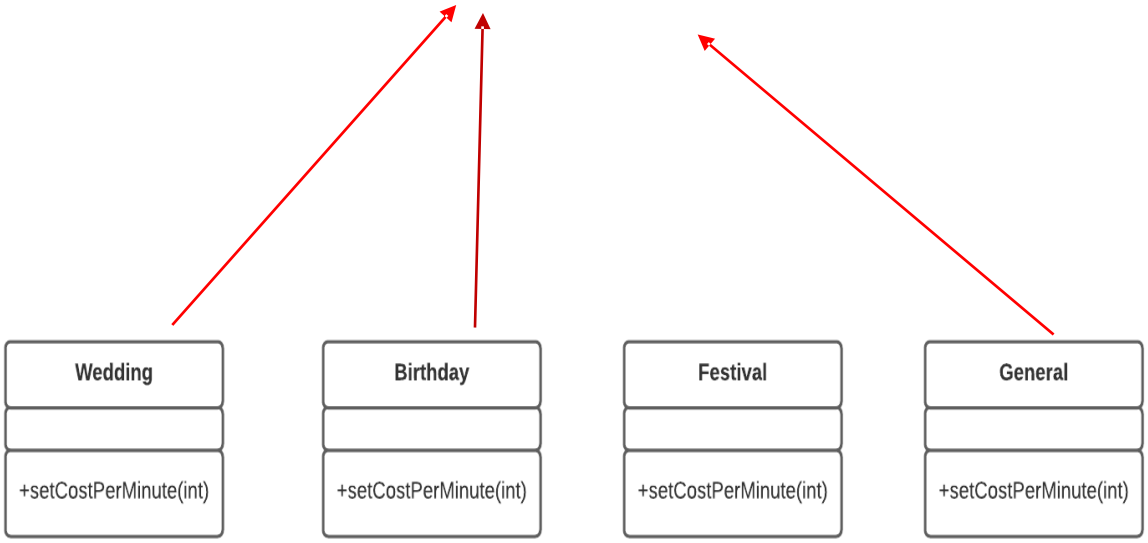


```
void  diskOut();
```

```
void Event::diskOut()
{
    char ch;
    ifstream read;
    read.open("E:\\Degree Courses\\Semester 3\\CS 212 -
    while (!(read.eof()))
    {
        ch = read.get();
        cout << ch;
    }
    read.close();
}
```

7. UML Diagram




Wedding
+setCostPerMinute(int)
Birthday
+setCostPerMinute(int)
Festival
+setCostPerMinute(int)
General
+setCostPerMinute(int)