COIS-2240H-A-F02-2024FA-PTBO
Software Design & Modelling
Lab 5
Jemima Ali
ID: 0652792
PART 1

```
0163.    //------------------------
0164.    // MEMBER VARIABLES
0165.    //------------------------
0166.
0167.    //Employee Attributes
0168.    private String name;
0169.    private int id;
0170.    private int salary;
0171.
0172.    //Employee Associations
0173.    private List<Shift> shifts;
0174.    private List<Privilege> privileges;
0175.    private Hospital hospital;
0176.    private List<Ward> wards;
0177.
0178.    //------------------------
0179.    // CONSTRUCTOR
0180.    //------------------------
0181.
0182.    public Employee(String aName, int aId, int aSalary, Hospital aHospital, Ward... allWards)
0183.    {
0184.      name = aName;
0185.      id = aId;
0186.      salary = aSalary;
0187.      shifts = new ArrayList<Shift>();
0188.      privileges = new ArrayList<Privilege>();
0189.      boolean didAddHospital = setHospital(aHospital);
0190.      if (!didAddHospital)
0191.      {
0192.        throw new RuntimeException("Unable to create employee due to hospital. See https://manual.umple.org?RE002ViolationofAssociationMultiplicity.html");
0193.      }
0194.      wards = new ArrayList<Ward>();
0195.      boolean didAddWards = setWards(allWards);
0196.      if (!didAddWards)
0197.      {
0198.        throw new RuntimeException("Unable to create Employee, must have at least 1 wards. See https://manual.umple.org?RE002ViolationofAssociationMultiplicity.html");
0199.      }
0200.    }
0201.
0202.    //------------------------
0203.    // INTERFACE
0204.    //------------------------
0205.
0206.    public boolean setName(String aName)
0207.    {
0208.      boolean wasSet = false;
0209.      name = aName;
```
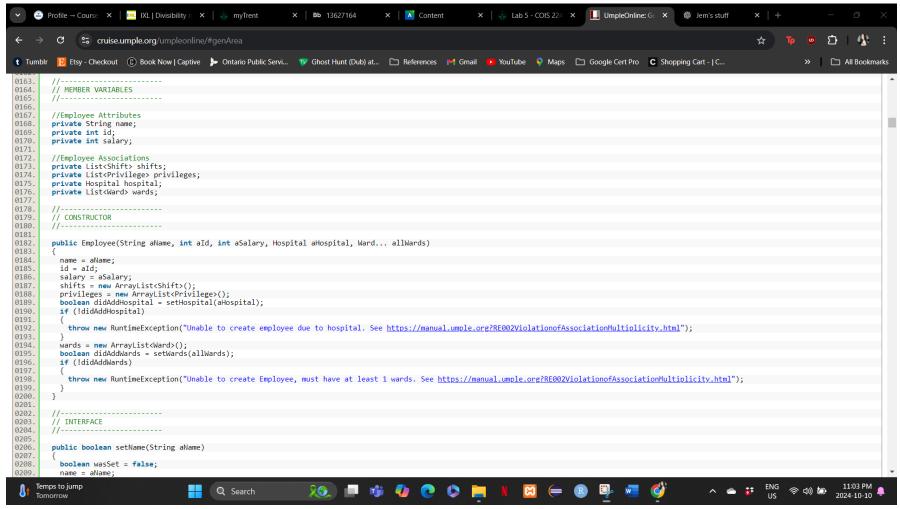
COIS-2240H-A-F02-2024FA-PTBO
Software Design & Modelling
Lab 5
Jemima Ali
ID: 0652792

```
// Hospital system - sample UML class diagram in Umple

class Hospital

{

    1 -- * Employee;

    1 -- 1..* Ward;

}


// Subsections within the hospital.

class Ward

{

    String name;

    Integer capacity;

    1..* -- * Employee;

    1 -- * Patient;

}


// The people who work at the hospital.

class Employee

{
```
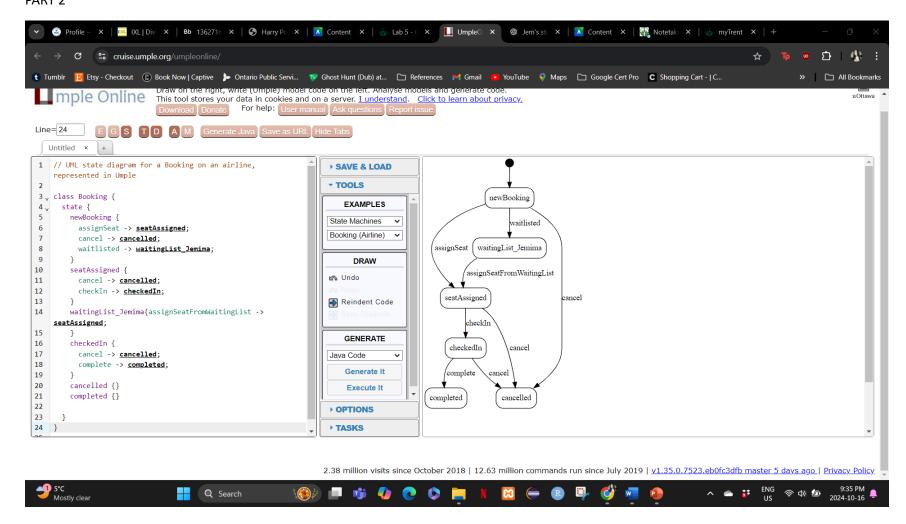
```
    String name;

    Integer id;

    Integer salary;

    1 -- 1..* Shift;

    1 -- 1..* Privilege;

}


// Working shifts that the employees have.

class Shift

{

    Date date;

    Integer startTime;

    Integer endTime;

}


// Various privileges and roles that the employees have.

class Privilege

{

    String privilege;
```

```
// Specialized employee who looks after patients.

class Doctor

{

    isA Employee;

}



// Specialized doctor who performs advanced procedures on patients.

class Surgeon

{

    isA Doctor;

}



// Employee who maintains the cleanliness of the hospital.

class Janitor

{

    isA Employee;

}
```

```
// Patient who is at the hospital to get better.

class Patient

{

    String name;

    * -- * Doctor;

}

class Inspector

{

 Jemima;

isA Employee;

 }


class Nurse

{

 isA Employee;

 * -- * Patient;

}
```

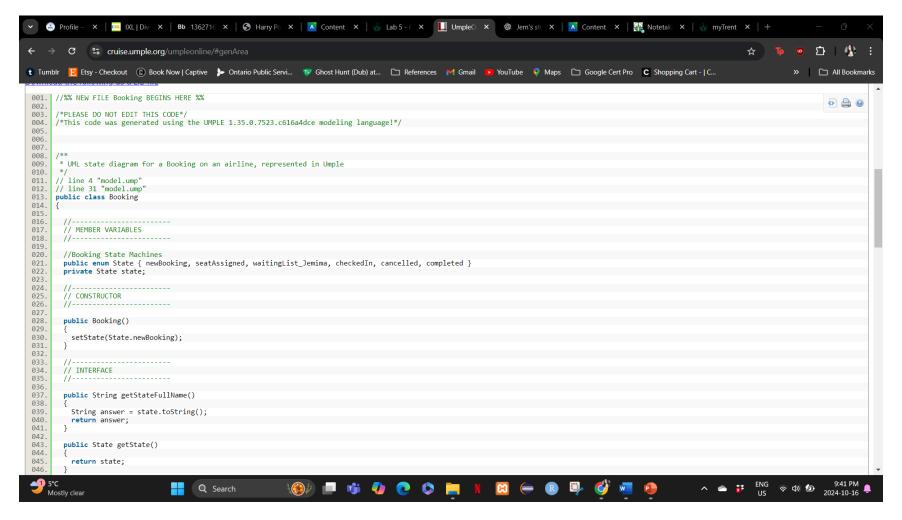COIS-2240H-A-F02-2024FA-PTBO
Software Design & Modelling
Lab 5
Jemima Ali
ID: 0652792
PART 2

COIS-2240H-A-F02-2024FA-PTBO
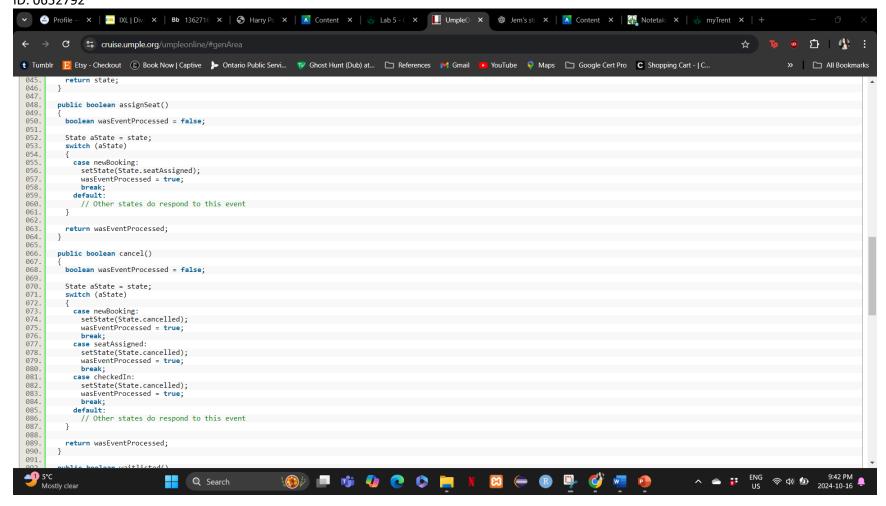Software Design & Modelling
Lab 5
Jemima Ali
ID: 0652792
PART 2: Java Code


```java
//%% NEW FILE Booking BEGINS HERE %%

/*PLEASE DO NOT EDIT THIS CODE*/
/*This code was generated using the UMPLE 1.35.0.7523.c616a4dce modeling language!*/


/**
 * UML state diagram for a Booking on an airline, represented in Umple
 */
// line 4 "model.ump"
// line 31 "model.ump"
public class Booking
{

  //------------------------
  // MEMBER VARIABLES
  //------------------------

  //Booking State Machines
  public enum State { newBooking, seatAssigned, waitingList_Jemima, checkedIn, cancelled, completed }
  private State state;

  //------------------------
  // CONSTRUCTOR
  //------------------------

  public Booking()
  {
    setState(State.newBooking);
  }

  //------------------------
  // INTERFACE
  //------------------------

  public String getStateFullName()
  {
    String answer = state.toString();
    return answer;
  }

  public State getState()
  {
    return state;
  }
}
```

```
045.        return state;
046.    }
047.
048.    public boolean assignSeat()
049.    {
050.        boolean wasEventProcessed = false;
051.
052.        State aState = state;
053.        switch (aState)
054.        {
055.            case newBooking:
056.                setState(State.seatAssigned);
057.                wasEventProcessed = true;
058.                break;
059.            default:
060.                // Other states do respond to this event
061.        }
062.
063.        return wasEventProcessed;
064.    }
065.
066.    public boolean cancel()
067.    {
068.        boolean wasEventProcessed = false;
069.
070.        State aState = state;
071.        switch (aState)
072.        {
073.            case newBooking:
074.                setState(State.cancelled);
075.                wasEventProcessed = true;
076.                break;
077.            case seatAssigned:
078.                setState(State.cancelled);
079.                wasEventProcessed = true;
080.                break;
081.            case checkedIn:
082.                setState(State.cancelled);
083.                wasEventProcessed = true;
084.                break;
085.            default:
086.                // Other states do respond to this event
087.        }
088.
089.        return wasEventProcessed;
090.    }
091.
092.    public boolean waitlisted()
```

```
090.    }
091.
092.    public boolean waitlisted()
093.    {
094.        boolean wasEventProcessed = false;
095.
096.        State aState = state;
097.        switch (aState)
098.        {
099.            case newBooking:
100.                setState(State.waitingList_Jemima);
101.                wasEventProcessed = true;
102.                break;
103.            default:
104.                // Other states do respond to this event
105.        }
106.
107.        return wasEventProcessed;
108.    }
109.
110.    public boolean checkIn()
111.    {
112.        boolean wasEventProcessed = false;
113.
114.        State aState = state;
115.        switch (aState)
116.        {
117.            case seatAssigned:
118.                setState(State.checkedIn);
119.                wasEventProcessed = true;
120.                break;
121.            default:
122.                // Other states do respond to this event
123.        }
124.
125.        return wasEventProcessed;
126.    }
127.
128.    public boolean assignSeatFromWaitingList()
129.    {
130.        boolean wasEventProcessed = false;
131.
132.        State aState = state;
133.        switch (aState)
134.        {
135.            case waitingList_Jemima:
136.                setState(State.seatAssigned);
```

COIS-2240H-A-F02-2024FA-PTBO
Software Design & Modelling
Lab 5
Jemima Ali
ID: 0652792

```java
132.        State aState = state;
133.        switch (aState)
134.        {
135.          case waitingList_Jemima:
136.            setState(State.seatAssigned);
137.            wasEventProcessed = true;
138.            break;
139.          default:
140.            // Other states do respond to this event
141.        }

143.        return wasEventProcessed;
144.      }

146.      public boolean complete()
147.      {
148.        boolean wasEventProcessed = false;

150.        State aState = state;
151.        switch (aState)
152.        {
153.          case checkedIn:
154.            setState(State.completed);
155.            wasEventProcessed = true;
156.            break;
157.          default:
158.            // Other states do respond to this event
159.        }

161.        return wasEventProcessed;
162.      }

164.      private void setState(State aState)
165.      {
166.        state = aState;
167.      }

169.      public void delete()
170.      {}

172.    }
```