```
001. //%% NEW FILE Booking BEGINS HERE %%
002.
003. /*PLEASE DO NOT EDIT THIS CODE*/
004. /*This code was generated using the UMPLE 1.29.1.4787.f023c4bb4 modeling
       language!*/
005.
006.
007.
008. /**
009.  * UML state diagram for a Booking on an airline, represented in Umple
010.  */
011. // line 4 "model.ump"
012. // line 31 "model.ump"
013. public class Booking
014. {
015.
016.   //------------------------
017.   // MEMBER VARIABLES
018.   //------------------------
019.
020.   //Booking State Machines
021.   public enum State { newBooking, seatAssigned, checkedIn, waitingList,
         cancelled, completed }
022.   private State state;
023.
024.   //------------------------
025.   // CONSTRUCTOR
026.   //------------------------
027.
028.   public Booking()
029.   {
030.     setState(State.newBooking);
031.   }
032.
033.   //------------------------
034.   // INTERFACE
035.   //------------------------
036.
037.   public String getStateFullName()
038.   {
039.     String answer = state.toString();
040.     return answer;
041.   }
042.
043.   public State getState()
044.   {
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()
093.   {
094.     boolean wasEventProcessed = false;
095.
096.     State aState = state;
097.     switch (aState)
098.     {
099.       case newBooking:
100.         setState(State.waitingList);
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 complete()
129.   {
130.     boolean wasEventProcessed = false;
131.
132.     State aState = state;
133.     switch (aState)
134.     {
135.       case checkedIn:
136.         setState(State.completed);
137.         wasEventProcessed = true;
138.         break;
139.       default:
140.         // Other states do respond to this event
141.     }
142.
143.     return wasEventProcessed;
144.   }
145.
146.   public boolean assignseattowaitingList()
147.   {
148.     boolean wasEventProcessed = false;
149.
150.     State aState = state;
151.     switch (aState)
152.     {
153.       case waitingList:
154.         setState(State.seatAssigned);
155.         wasEventProcessed = true;
156.         break;
157.       default:
158.         // Other states do respond to this event
159.     }
160.
161.     return wasEventProcessed;
162.   }
163.
164.   private void setState(State aState)
165.   {
166.     state = aState;
167.   }
168.
```

```
169.   public void delete()
170.   {}
171.
172. }
```