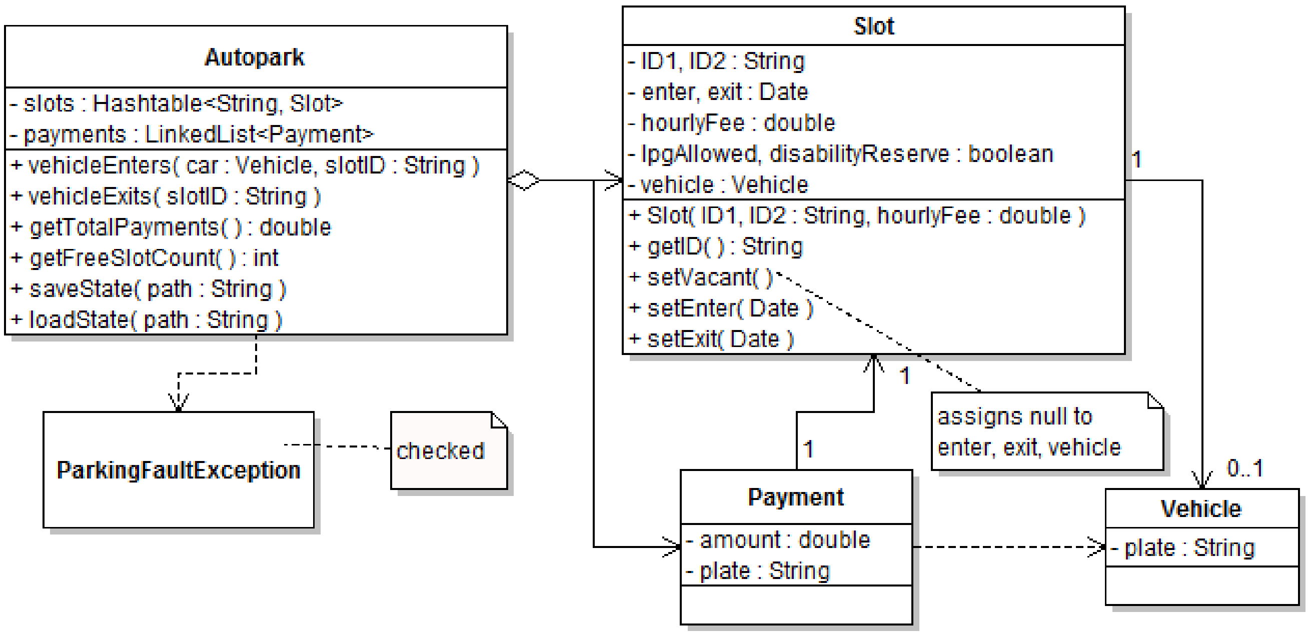
**BLM2562 OBJECT ORIENTED PROG., Section 2, FINAL EXAM** **(01/06/2015)**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Duration:** | 90 mins. | | **Score:** | | | | | | **Student Nr:** | **Signature:** |
| **Grading:** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **Name, Surname:** | |
| | 10 | 10 | 15 | 15 | 50 | | | | | |

## QUESTIONS



Autopark
- slots : Hashtable<String, Slot>
- payments : LinkedList<Payment>
+ vehicleEnters( car : Vehicle, slotID : String )
+ vehicleExits( slotID : String )
+ getTotalPayments( ) : double
+ getFreeSlotCount( ) : int
+ saveState( path : String )
+ loadState( path : String )
Slot
- ID1, ID2 : String
- enter, exit : Date
- hourlyFee : double
- lpgAllowed, disabilityReserve : boolean
- vehicle : Vehicle
+ Slot( ID1, ID2 : String, hourlyFee : double )
+ getID( ) : String
+ setVacant( )
+ setEnter( Date )
+ setExit( Date )
1
1
1
0..1
assigns null to enter, exit, vehicle
ParkingFaultException
checked
Payment
- amount : double
- plate : String
Vehicle
- plate : String

Answer these questions according to the UML class schema given above. You may need to extract hidden information from the schema and add necessary code while answering the questions. You don't have to consider inconsistencies other than the ones related with the exceptions in the class schema. You will be writing some part of an autopark information system.

**Question 1:** Write the source code of ParkingFaultException.

**Question 2:** Write the source code of class Vehicle. Some vehicles are fuelled by LPG and/or some vehicles are used by drivers having a disability.

**Question 3:** Write the source code of the setExit method of the class Slot. You do not need to call any other methods of this class when coding. The exit time of a vehicle from a slot cannot be earlier than its entrance time. Otherwise, a ParkingFaultException must be generated.

**Question 4:** Write the source code of class Payment. You can use the following method in order to calculate the parking fee:

- `public long Date.getTime()`: Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this `Date` object.

**Question 5:** Write the source code of only the following methods of the class Autopark. Prevent simultaneous calling of a method while coding it.

- vehicleExits: This method collects the parking fee and then empties the slot. If a non-existent slot's ID has been passed as parameter, a ParkingFaultException must be generated.
- getTotalPayments: Returns the total amount of fees collected so far.
- getFreeSlotCount: Returns the number of the free slots.
- saveState: Saves the slots and payments to the given file.

**BLM2562 OBJECT ORIENTED PROG., Section 2, FINAL EXAM** **(01/06/2015)**

**Question 1:** Write the source code of ParkingFaultException.

```
@SuppressWarnings("serial")
public class ParkingFaultException extends java.io.IOException {
   public ParkingFaultException(String arg0) {
      super(arg0);
   }
}
```

**Question 2:** Write the source code of class Vehicle.

```
public class Vehicle implements java.io.Serializable {
   private static final long serialVersionUID = 1L;
   private String plate;
   private boolean hasLPG, disabledOwner;
   public Vehicle(String plate) {
      this.plate = plate;
   }
   public boolean isLPG() { return hasLPG; }
   public void setLPG(boolean hasLPG) { this.hasLPG = hasLPG; }
   public boolean isDisabled() { return disabledOwner; }
   public void setDisabledOwner(boolean disabledOwner) {
      this.disabledOwner = disabledOwner;
   }
   public String getPlate() { return plate; }
}
```

**Question 3:** Write the source code of the setExit method of the class Slot.

```
   public void setExit(Date exit) throws ParkingFaultException {
      if( exit.after(enter) )
         this.exit = exit;
      else
         throw new ParkingFaultException("Inconsistent exit date");
   }
```

**Question 4:** Write the source code of class Payment.

```
public class Payment implements java.io.Serializable {
   private static final long serialVersionUID = 1L;
   private double amount;
   private String plate;
   public Payment(Slot slot) {
      this.plate = slot.getVehicle().getPlate();
      long hours = slot.getExit().getTime()-slot.getEnter().getTime();
      amount = slot.getHourlyFee() * (int)(hours/(1000*60*60));
   }
   public double getAmount() { return amount; }
   public String getPlate() { return plate; }
}
```

**Question 5:** Write the source code of the following methods of the class Autopark:

```java
public synchronized void vehicleExits( String slotID ) throws ParkingFaultException {
   Slot aSlot = slots.get(slotID);
   if( aSlot == null )
      throw new ParkingFaultException("UndefinedSlot: "+slotID);
   aSlot.setExit( new Date() );
   payments.add( new Payment(aSlot) );
   aSlot.setVacant();
}
public synchronized double getTotalPayments( ) {
   double sum = 0;
   for (Payment aPayment : payments)
      sum += aPayment.getAmount();
   return sum;
}
public synchronized int getFreeSlotCount( ) {
   int count = 0;
   for( Slot aSlot : slots.values() )
      if( aSlot.getVehicle() == null )
         count++;
   /*if coded, isVacant method can also be used for counting*/
   return count;
}
public synchronized void saveState( String aFile ) {
   try {
      ObjectOutputStream str = new ObjectOutputStream( new FileOutputStream(aFile) );
      str.writeObject(slots);
      str.writeObject(payments);
      str.close();
   }
   catch (IOException e) {
      e.printStackTrace();
   }
}
```