Oppgave 2.4

Det er tre feil i koden. Full uttelling er 6 poeng. 1 poeng for hvert linjenummer som er riktig og 1 poeng for hver riktige kodelinje som blir foreslått erstattet. Det følgende er løsningsforslaget:

```
// line 25
alarm = new Alarm();

// line 29
return alarm;

// line 39
house.getAlarm().soundTheAlarm();
```

Oppgave 3.1.1

```java
public class Bus_Oppgave3_1_1 {
    private String id;
    private String modelName;
    private int maxCapacity;
    private Route route;
    private TicketAuthenticator ticketAuthenticator;

    public Bus_Oppgave3_1_1(String id, String modelName, int maxCapacity) {
        this.id = id;
        this.modelName = modelName;
        this.maxCapacity = maxCapacity;
        this.route = null;
        this.ticketAuthenticator = null;
    }

    public Bus_Oppgave3_1_1(String id, int maxCapacity, String modelName,
                            Route route, TicketAuthenticator ticketAuthenticator) {
        this.id = id;
        this.modelName = modelName;
        this.route = route;
        this.ticketAuthenticator = ticketAuthenticator;
        this.maxCapacity = maxCapacity;
    }

    @Override
    public String toString() {
        return "Bus model: " + modelName + ", Number of seats: " + maxCapacity;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```

Oppgave 3.1.2

```java
public class CollectiveTransportUnit {
    private String id;
    private int maxCapacity;

    public CollectiveTransportUnit(String id, int maxCapacity) {
        this.id = id;
        this.maxCapacity = maxCapacity;
    }

    // Gettere og settere kan utelates
    public String getId() {
        return id;
    }

    public void setId(String id) { this.id = id; }

    public int getMaxCapacity() { return maxCapacity; }

    public void setMaxCapacity(int maxCapacity) { this.maxCapacity = maxCapacity; }
}
```

Merk at man nå må benytte getMaxCapacity() i toString(). Pass på at super benyttes.

```java
public class Bus_Oppgave_3_1_2 extends CollectiveTransportUnit{
    private String modelName;
    private Route route;
    private TicketAuthenticator ticketAuthenticator;

    public Bus_Oppgave_3_1_2(String id, String modelName, int maxCapacity) {
        super(id, maxCapacity);
        this.modelName = modelName;
        this.route = null;
        this.ticketAuthenticator = null;
    }

    public Bus_Oppgave_3_1_2(String id, int maxCapacity, String modelName,
                             Route route, TicketAuthenticator ticketAuthenticator) {
        super(id, maxCapacity);
        this.modelName = modelName;
        this.route = route;
        this.ticketAuthenticator = ticketAuthenticator;
    }

    @Override
    public String toString() {
        return "Bus model: " + modelName + ", Number of seats: " + getMaxCapacity();
    }

}
```

Oppgave 3.2

Merk hvordan for-løkken defineres (Teller ikke med siste element).

```
public class Route {
    private ArrayList<BusStop> stops = new ArrayList<>();

    public double calculateRouteLength() {
        int numberOfBusStops = stops.size();

        double totalDistance = 0;
        for (int x = 0; x < numberOfBusStops-1; x++) {
            BusStop currentBusStop = stops.get(x);
            BusStop nextBusStop = stops.get(x+1);
            totalDistance += GPSService.distanceBetweenBusStops(currentBusStop, nextBusStop);

        }

        return totalDistance;
    }
}
```

Oppgave 3.3

Merk at "return false" ikke trenger å være med for å få full uttelling

```
public class Bus_Oppgave_3_3 implements TicketValidation {
    @Override
    public boolean checkTicketValidity(Ticket ticket) {
        return false;
    }
}
```

Oppgave 3.4

Merk at man må caste ticket til RouteTicket for å hente ruten.

```java
public boolean checkTicketValidity(Ticket ticket) {
    boolean ticketIsAuthentic = ticketAuthenticator.checkTicketAuthenticity(ticket);
    if (ticketIsAuthentic) {
        if (ticket instanceof PremiumTicket) {
            System.out.println("The ticket is valid!");
            return true;
        } else if (ticket instanceof RouteTicket &&
                ((RouteTicket) ticket).getRoute() == route) {
            System.out.println("The ticket is valid!");
            return true;
        }
    }
    System.out.println("The ticket is not valid!");
    return false;
}
```

Oppgave 3.5

```java
public class TicketAuthenticator {

    public boolean checkTicketAuthenticity(Ticket ticket) {
        try (DatabaseConnection db = new DatabaseConnection()) {
            db.openConnection();
            return db.checkTicket(ticket);
        }
        catch(DatabaseException dbException) {
            System.out.println("Oops! Something went wrong with the database communication.");
            return false;
        }
    }
}
```