OPMS 2018 - Exercise 6

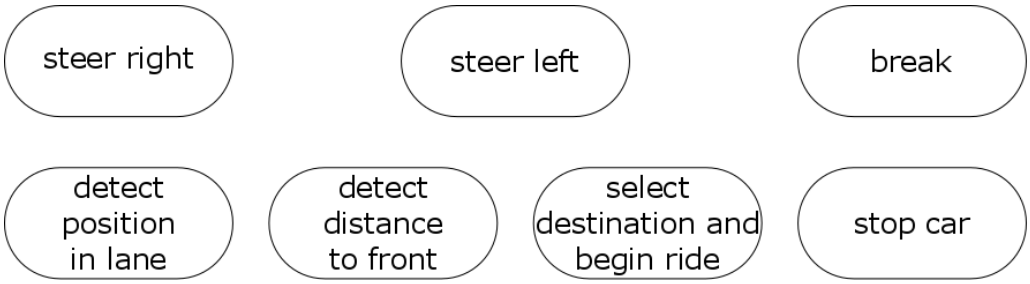
Task 1 (Activity Diagram)

Create an activity diagram for an autonomous car using the following process description and specified activities. There is space on the next page to create the diagram.

Use each specified activity exactly once!

- First of all a destination has to be selected and the ride begins.
- Next, the car detects the distance to other cars in the front as well as its position in lane. If necessary corrections are made:
  - If the distance is smaller than the safety distance, the car has to break.
  - If the relative position in the lane is larger than 5 cm, the car has to steer left or right, depending on the direction of the deviation.

  Both processes, the distance to other cars as well as the position in the lane run in parallel.
- If the destination has not been reached, the previous step is repeated.
- Eventually, after the destination has been reached, the car is stopped.



Activity Diagram:

Task 2 (UML)

Draw the Class Diagram for the following code

```
public abstract class Construction {
    protected int yearOfConstruction;
    protected double length;
    protected double width;
    protected double height;

    public int getYearOfConstruction() {
        return yearOfConstruction;
    }
}
```

```
public class Building extends Construction {
    protected int numberOfRooms;
    protected int numberOfFloors;

    public int getNumberOfRooms() {
        return numberOfRooms;
    }

    public double getFloorArea() {
        return length*width*numberOfFloors;
    }
}
```

You can use this space (or the back of this page) to draw the Class Diagram:

Task 3 (UML)

In this task, you will implement an automatic teller machine (ATM).

1. In Eclipse, create a new Java project named “Exercise6”. Right-click the “src”-folder and create a new package called “atm”. Create all classes of this exercise in this package.
2. Implement the class `Account` as specified by the following class diagram.

Account
-pin: int -balance: double
+Account(pin: int, balance: double) +pinIsValid(int pin): boolean +getBalance(): double +setBalance(newBalance: double): void

Figure 1: ATM (Valera N. Trubin)

3. Create a new class `Atm` and add a main()-method. In the main()-method, instantiate a HashMap `accounts` where the keys are Integers and the values are Accounts. This Map will serve to find an account that corresponds to a card number.
   Hint: Don’t forget to import Map and HashMap from java.util.

4. Fill the `accounts`-Map with three sample accounts.
   Hint: Use put() to add an entry to the map.

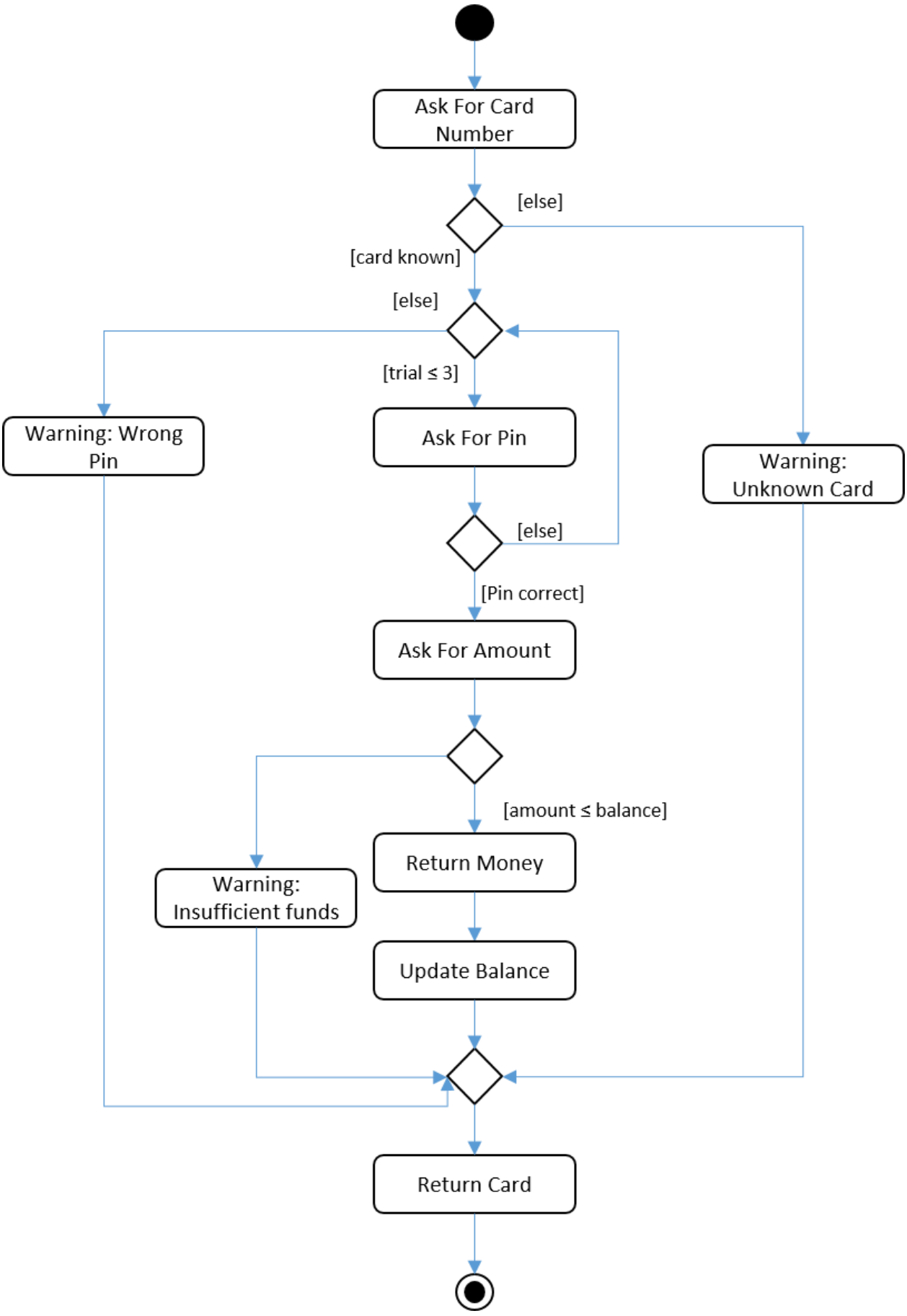
5. Implement the functionality described by the activity diagram on the next page. Use an int to count the PIN entry trials (maximum = 3).

   Hint: A map’s get() function will return null, if no element was found for the given key.
   Hint: To get user input, import and use the java.util.Scanner module.

```
Scanner reader = new Scanner(System.in);
int userInput1 = reader.nextInt();
double userInput2 = reader.nextDouble();
```

   Hint: If you use a loop to count the PIN entry trials, you can use `break;` to stop the loop earlier than defined. For example, the following code not run multiple times, but only once.

```
50    for (int x = 0; x <= 5; x++) {
51        // Your code here
52        break;
53    }
```


Ask For Card Number
[else]
[card known]
[else]
[trial ≤ 3]
Warning: Wrong Pin
Ask For Pin
Warning: Unknown Card
[else]
[Pin correct]
Ask For Amount
[amount ≤ balance]
Return Money
Warning: Insufficient funds
Update Balance
Return Card