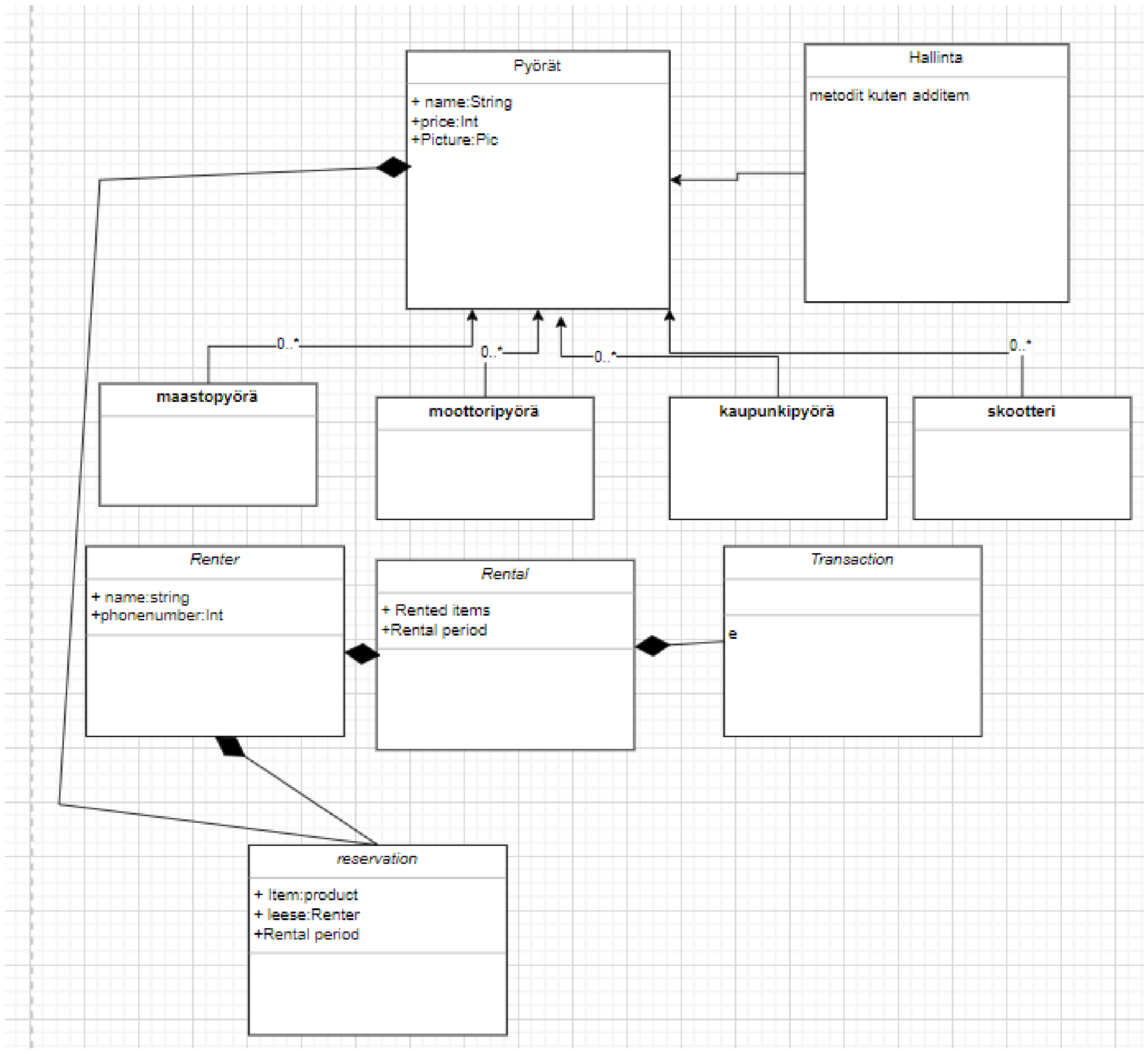
## Class structure



**User case**

I click on the "Add New Bike" button and enter  the name of the new bike, the price per hour, and uploads some pictures of the bike. I set the availability of the bike, which can be either "available" or "unavailable" and select the bike type, which can be either "mountain bike", "road bike", "city bike", or "electric bike".
The program confirms that the new bike has been added successfully and displays the updated list of bikes.

My customer, Kaarlo, wants to rent a road bike. He searches for the road bikes and checks their availability. He sees that there are two road bikes available for rental for the desired week. He enters his name and phone number as the renter's information.
He selects the rental period as one week and the pricing option as weekly.
The program calculates the rental fee as 7 x 100 = 700 euros and displays the fee to Kaarlo.
Kaarlo confirms the transaction and  the program marks the road bike as rented for the desired week and updates the number of available road bikes.
The program displays the updated rental list, which now includes Kaarlo’s rental of the road bike.

The program displays the summary of income and expenses for the current month, which includes the rental fees, purchase costs, storage costs, and repair costs.
I can see the total income for each bike type and the total income for all bikes.
I can also see the total expenses for each bike type and the total expenses for all bikes.

**Algorithm**

1. Create a class for each bike, containing attributes such as name, price, and availability.
2. For each bike, determine the hourly, daily, and weekly rental prices. Use a formula to calculate the discounted price for longer rental periods.
3. Create a class for each renter, containing attributes such as name and contact information.
4. Create a reservation system using a calendar. Each reservation should be linked to the specific bike being rented.
5. When a bike is rented, mark its availability in the system and assign it to the renter.
6. When a bike is returned, update its availability in the system and calculate the rental fee based on the rental period and the pricing formula.
7. If a bike is damaged or lost, record it in the system and calculate a refund or fee based on the situation.
8. Generate views and summaries that show the demand and income for each bike, as well as the income for each bike type over a desired period.
9. Allow the lessor to add or remove bikes from the system as needed.
10. Keep a record of each rental, including the bike rented, the renter, the rental period, and any associated comments.

Data structures
Item Class: This class could be used to store the properties of each rental item, such as its name, price, picture, and availability. Each instance of the class represents a single item, and it can also store information about the item's rental history and any damages/repairs.

Rental Class: This class could be used to store information about each rental transaction, including the renter's name and contact information, the rented item(s), the rental period, and any comments. Each instance of the class represents a single rental transaction, and it can also track the rental fees and any damages/repairs incurred during the rental period.

Reservation Class: This class could be used to store information about each advance reservation, including the reserved item(s), the reservation period, and the customer's contact information. Each instance of the class represents a single reservation, and it can also track whether the reservation has been confirmed or cancelled.

Availability Schedule: This data structure could be used to track the availability of each rental item over time, so that the program can determine when an item will be available for rent or when a reservation can be confirmed. This could be implemented as a matrix, where the rows represent the rental items and the columns represent time periods (e.g., hours, days, or weeks).

Hash Map: A hash map could be used to store all the items in the inventory, using their names as keys. This would allow for efficient lookups when adding new rentals or reservations, as well as when searching for an item in the inventory.

Linked List: A linked list could be used to store the rental history and any damages/repairs for each item. This would allow for efficient insertion and deletion of rental records, as well as the ability to traverse the rental history in chronological order.