Design task: Logistics

Software engineering
Szoftvertechnológia

Dr. Balázs Simon
BME, IIT

Outline

- Requirements for Logistics
- Class diagram
- Sequence diagrams
- Alternative designs

Requirements: Logistics

Requirements

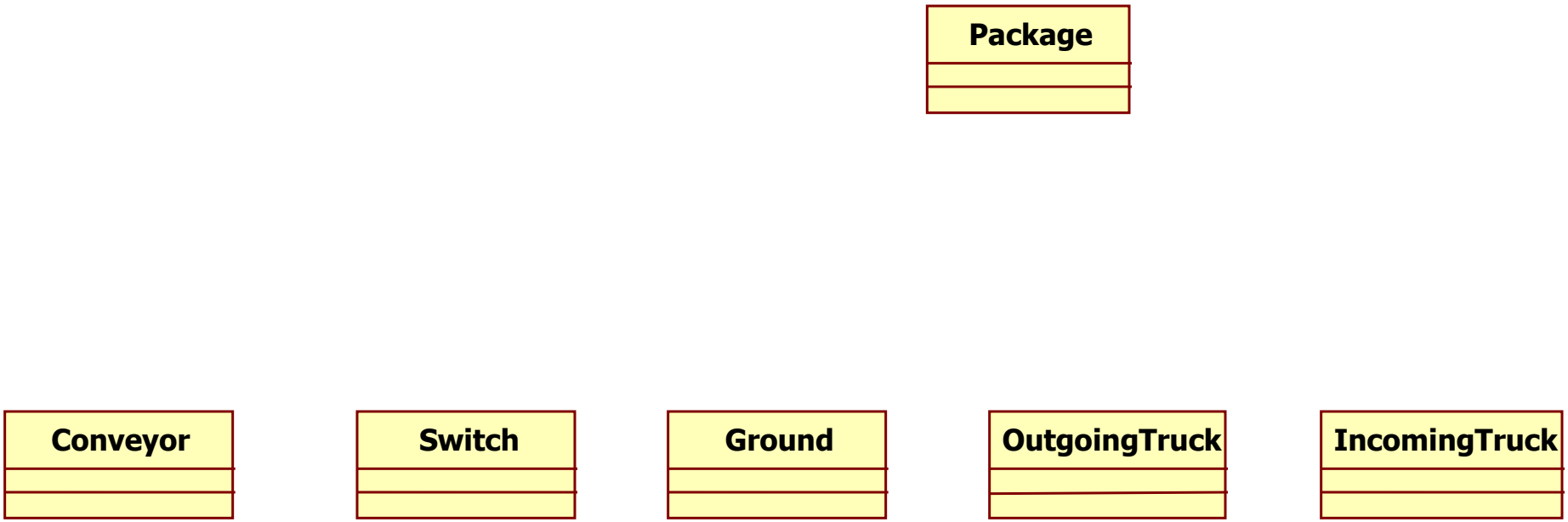
- In a package distributing station the packages are coming in and leaving in trucks. The packages are transferred between trucks using conveyor belts. The task of the player is to control the system of conveyor belts.
- Trucks bring packages to entry points and take away packages from exit points. A package has a color and a spoil time. A package with a given color must be forwarded to a truck expecting a package of the same color. If the package is perishable (not every package is), it must arrive to the truck before it is spoiled. If the package does not arrive in time, it explodes.
- A truck at an entry point puts packages to the conveyor belt starting there at random times. The package moves on the conveyor belt and at the end it falls off. It can fall to the following places:
  - somewhere onto another conveyor belt
  - into a truck at an exit point
  - to the ground if there is nothing at the end
  - to a switch

Requirements

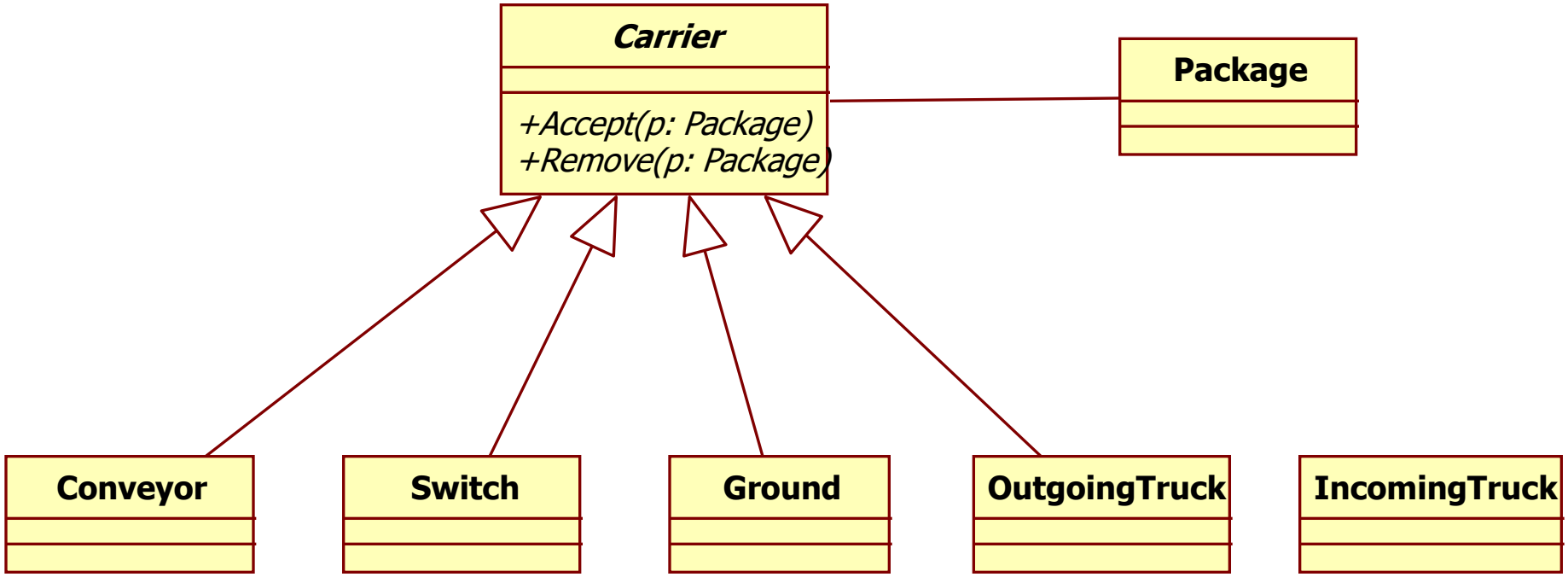
- The switch can change the direction of the package. There are always one or more incoming conveyor belts and always two or more outgoing conveyor belts at a switch. The package spends some time on the switch, then it moves forward to the conveyor belt pointed by the switch. The player can change the direction of the switch.
- The player gets points for every correctly delivered package. A package is correctly delivered if it arrives to a truck waiting for a package of the same color. If the package arrives to the wrong truck, the player loses points. The player also loses points if a package explodes. If a package falls to the ground or two packages are smashed together, they break. Breaking packages is a serious mistake.
- The colors and spoiling times of the packages coming from the incoming trucks are unknown for the player in advance. The delivery plan of the outgoing trucks is also unknown for the player in advance. The delivery plan prescribes the order and colors of the packages to be packed into the truck.

Design steps for the Class Diagram

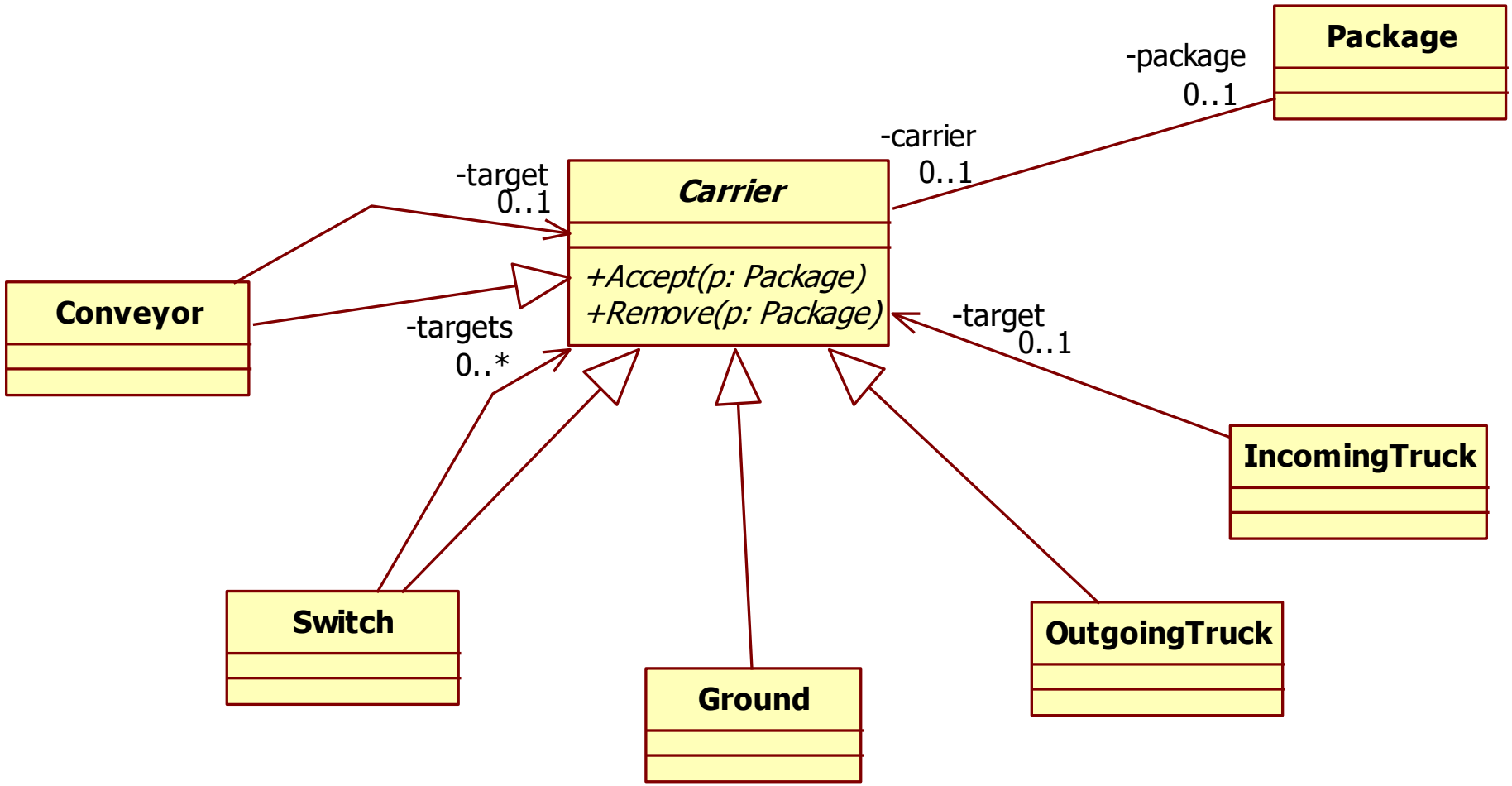
Essential classes



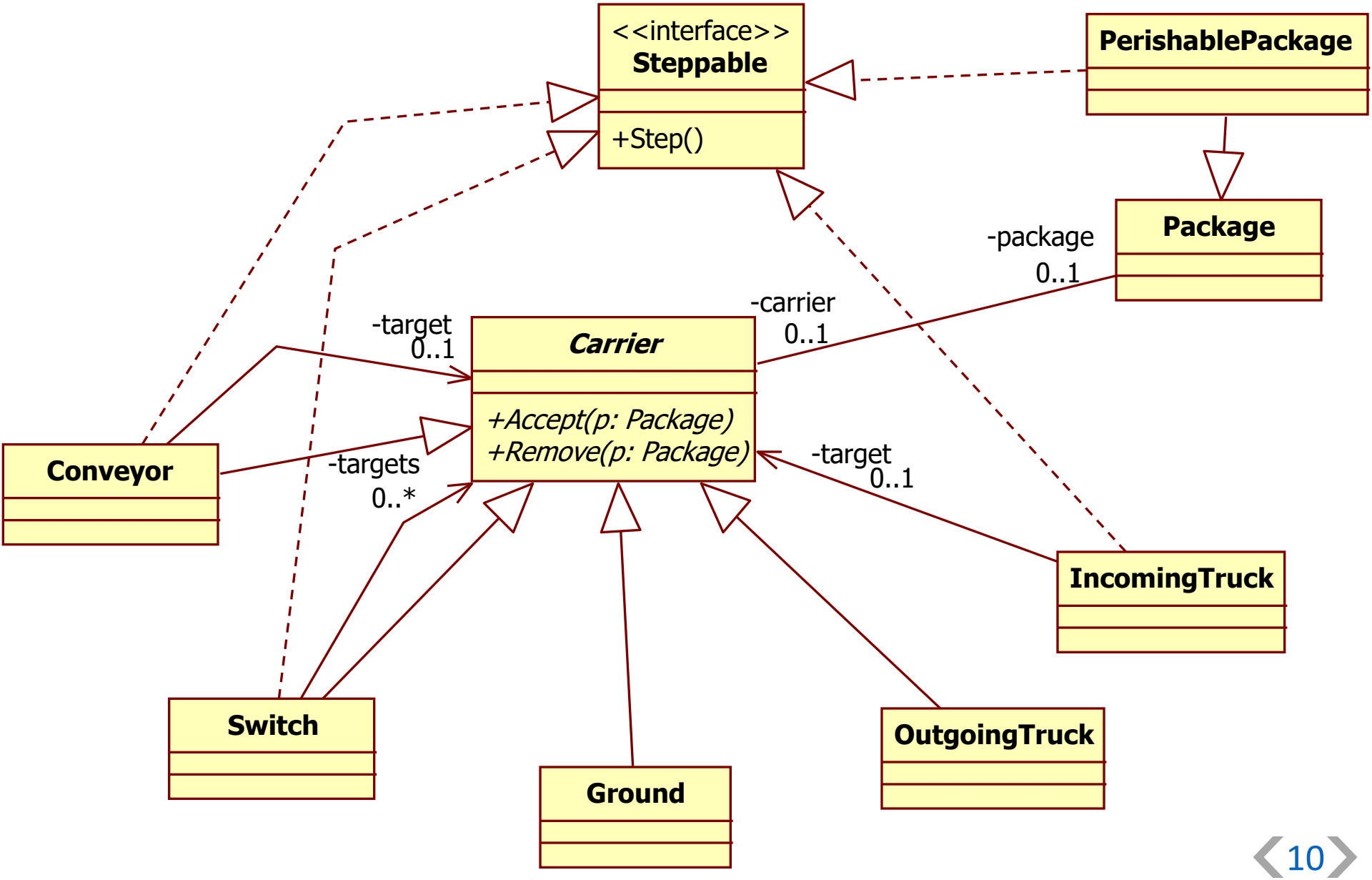
Common ancestor, connection to the package



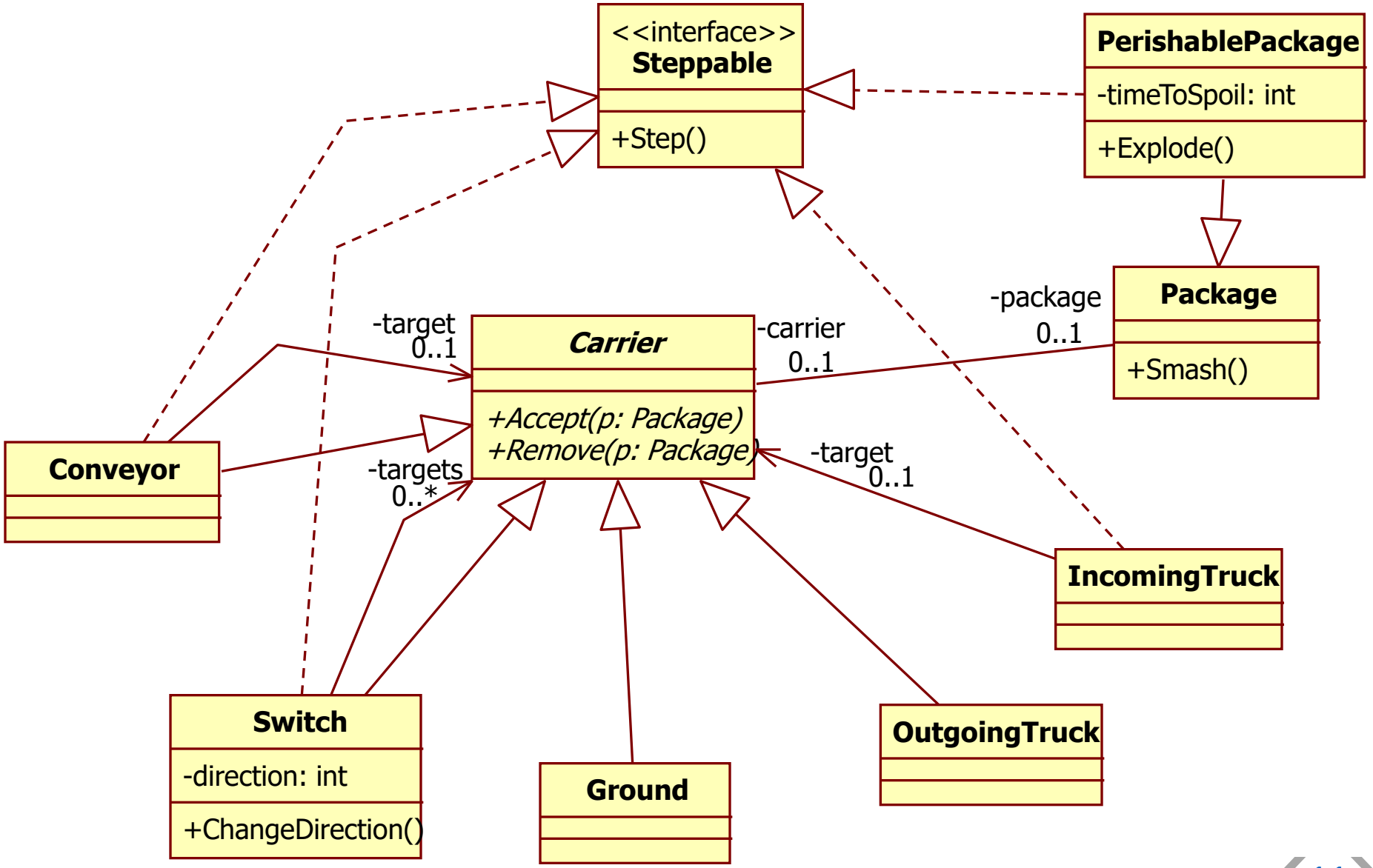
Next carrier



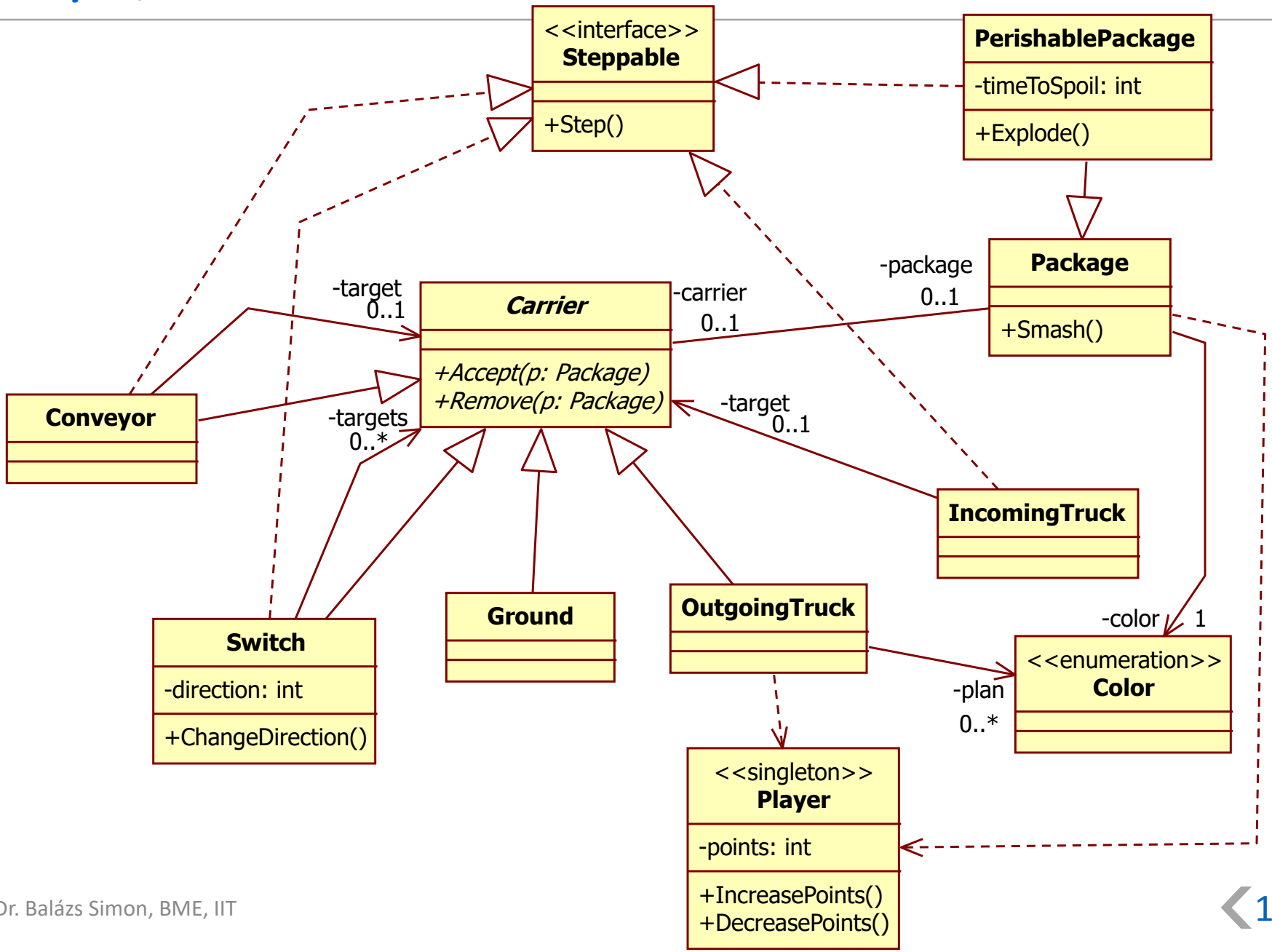
Steppable elements



Rest of the methods and fields

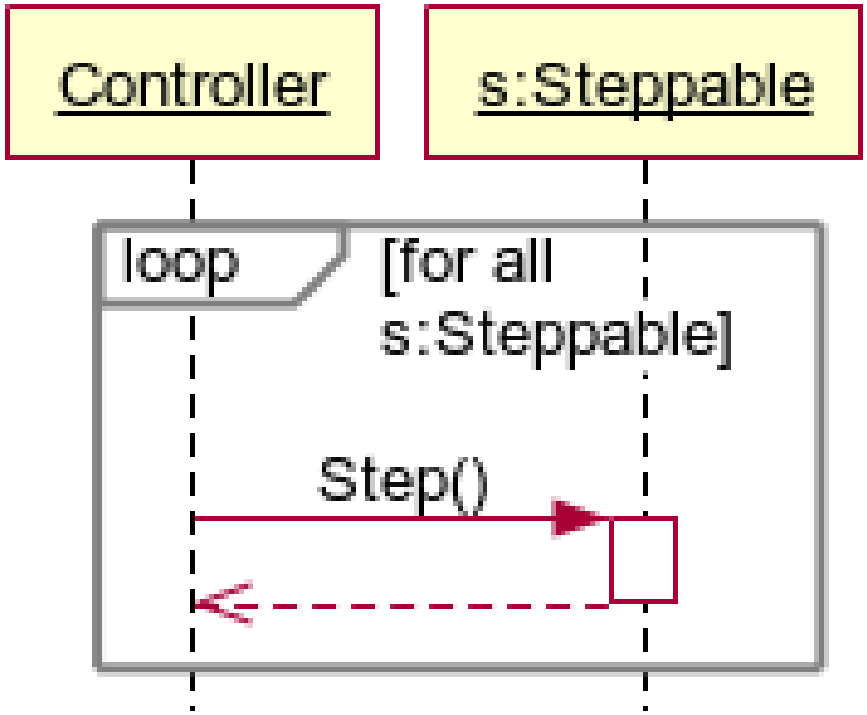


Player, Color

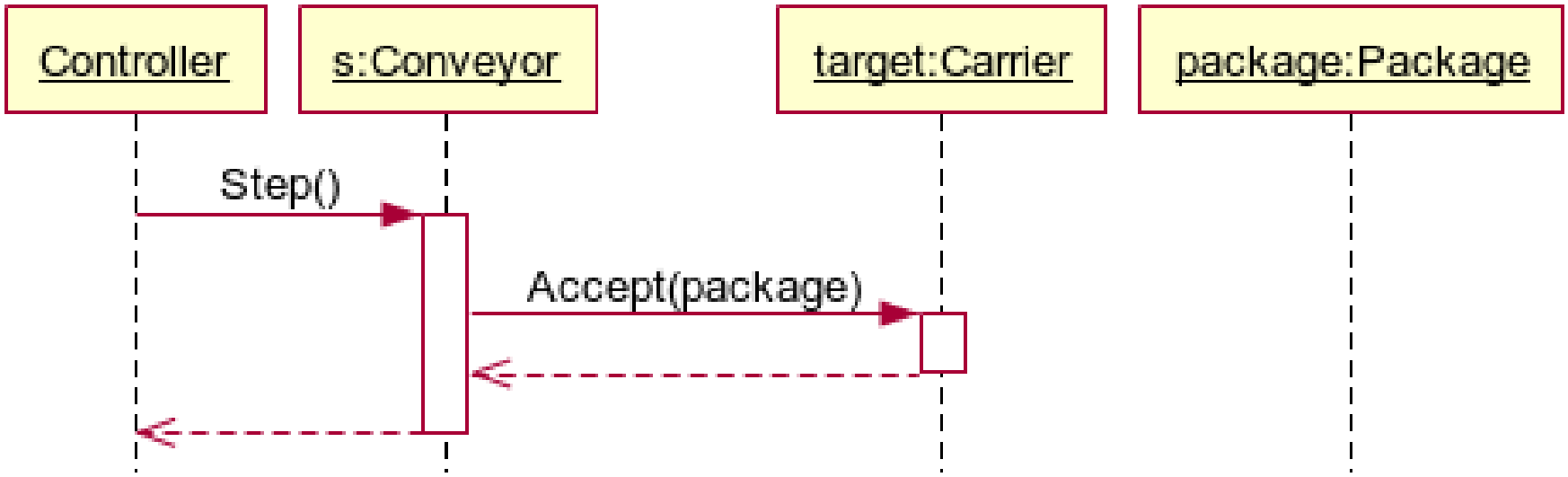


Sequence Diagrams

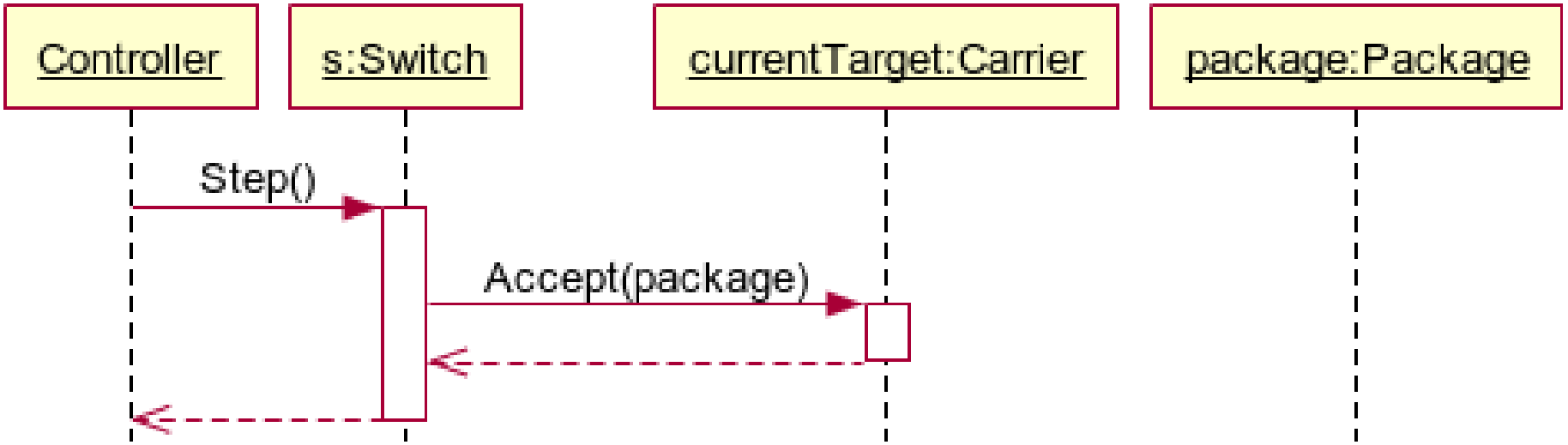
Making elements step



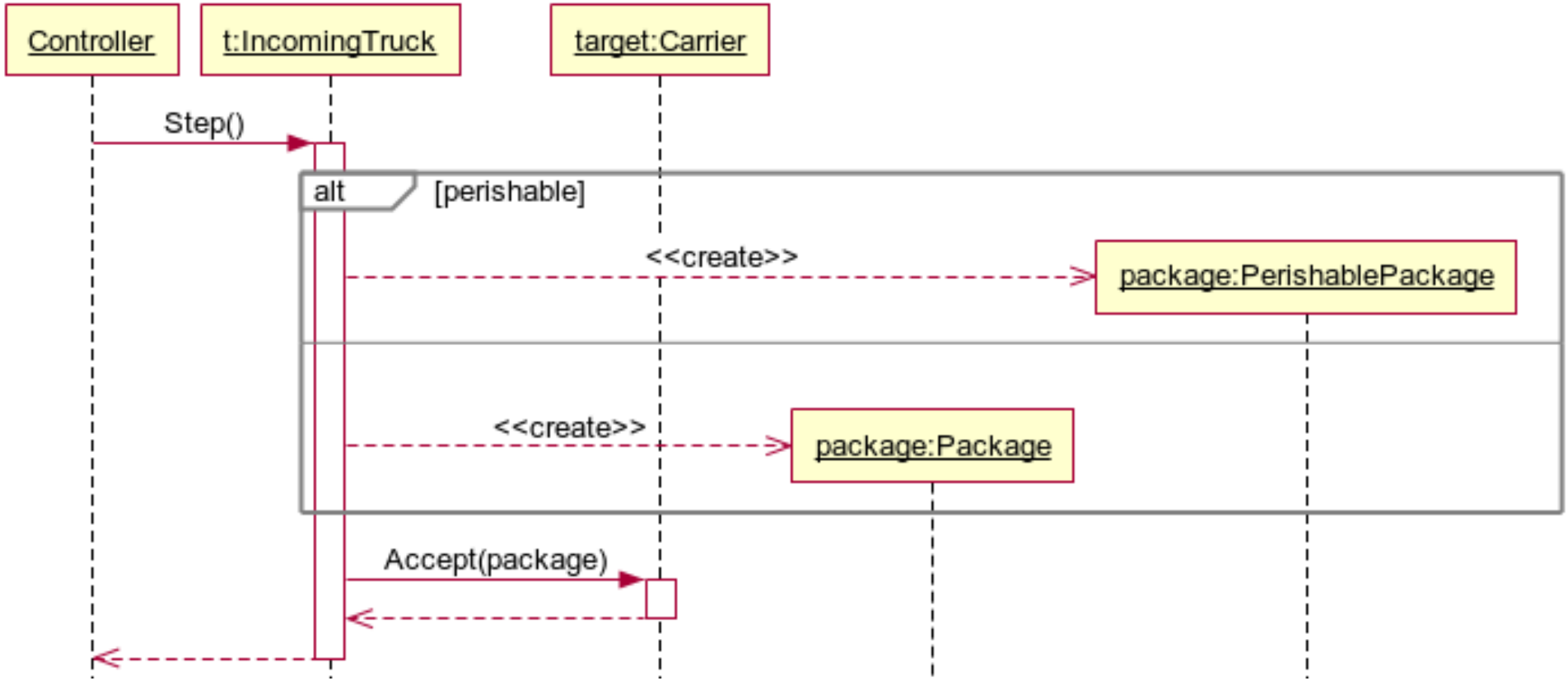
Conveyor steps



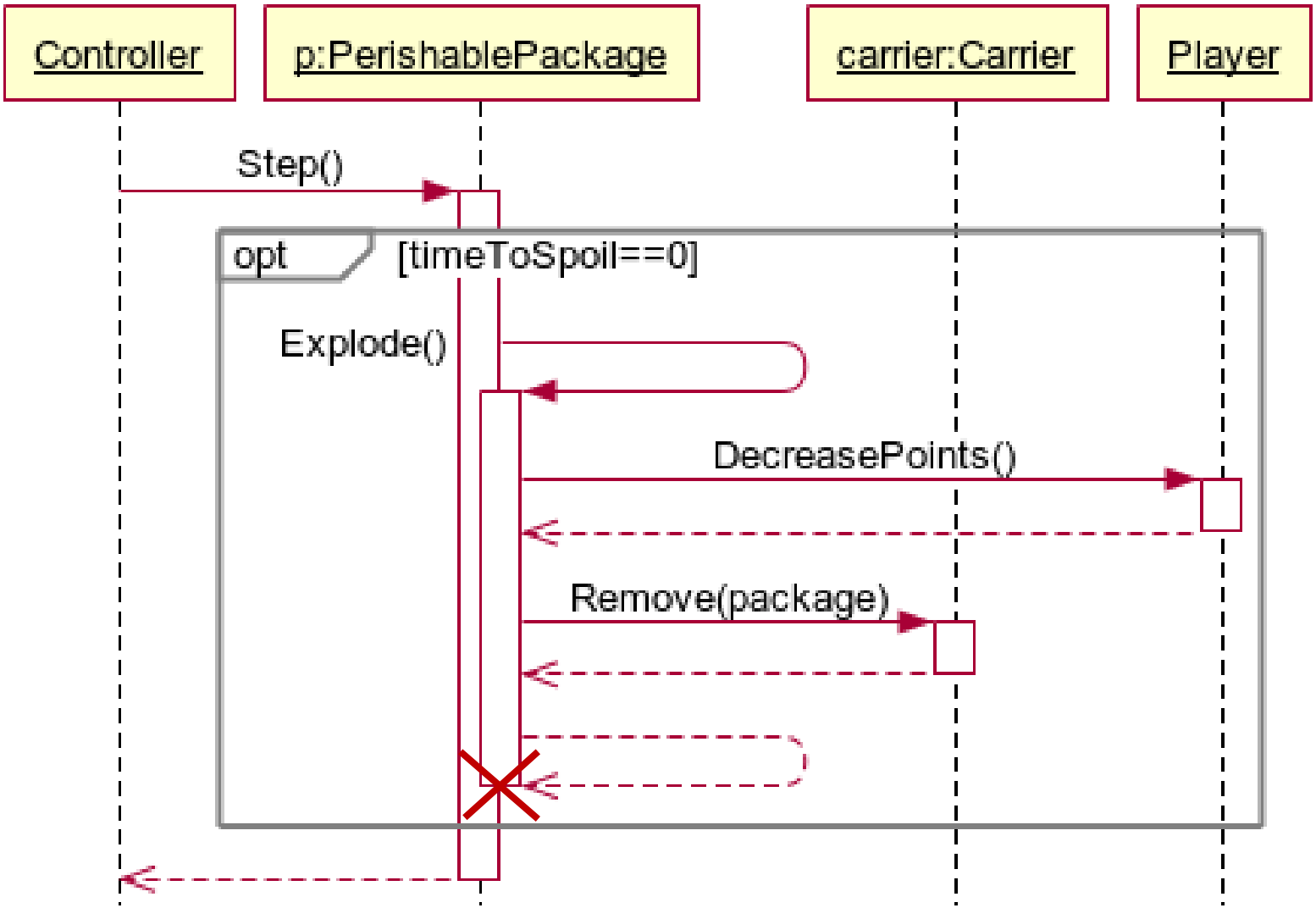
Switch steps



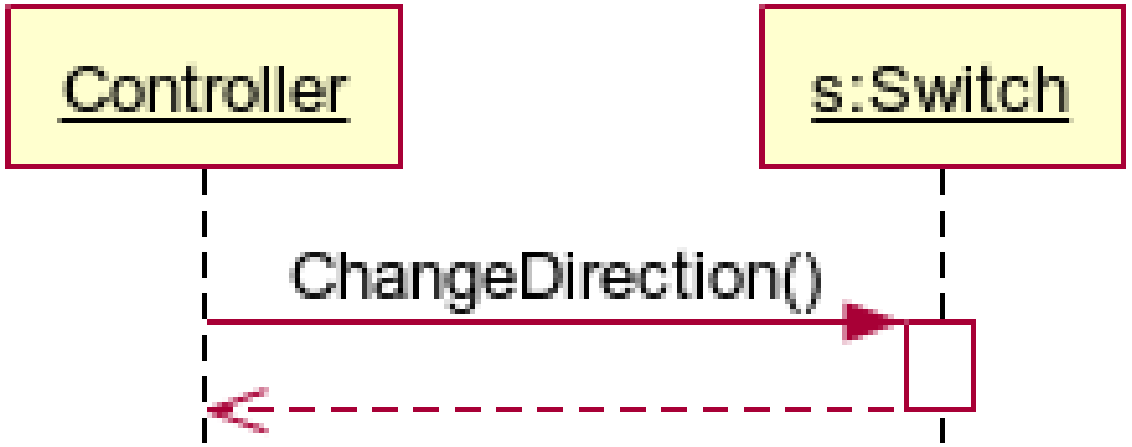
Incoming truck steps



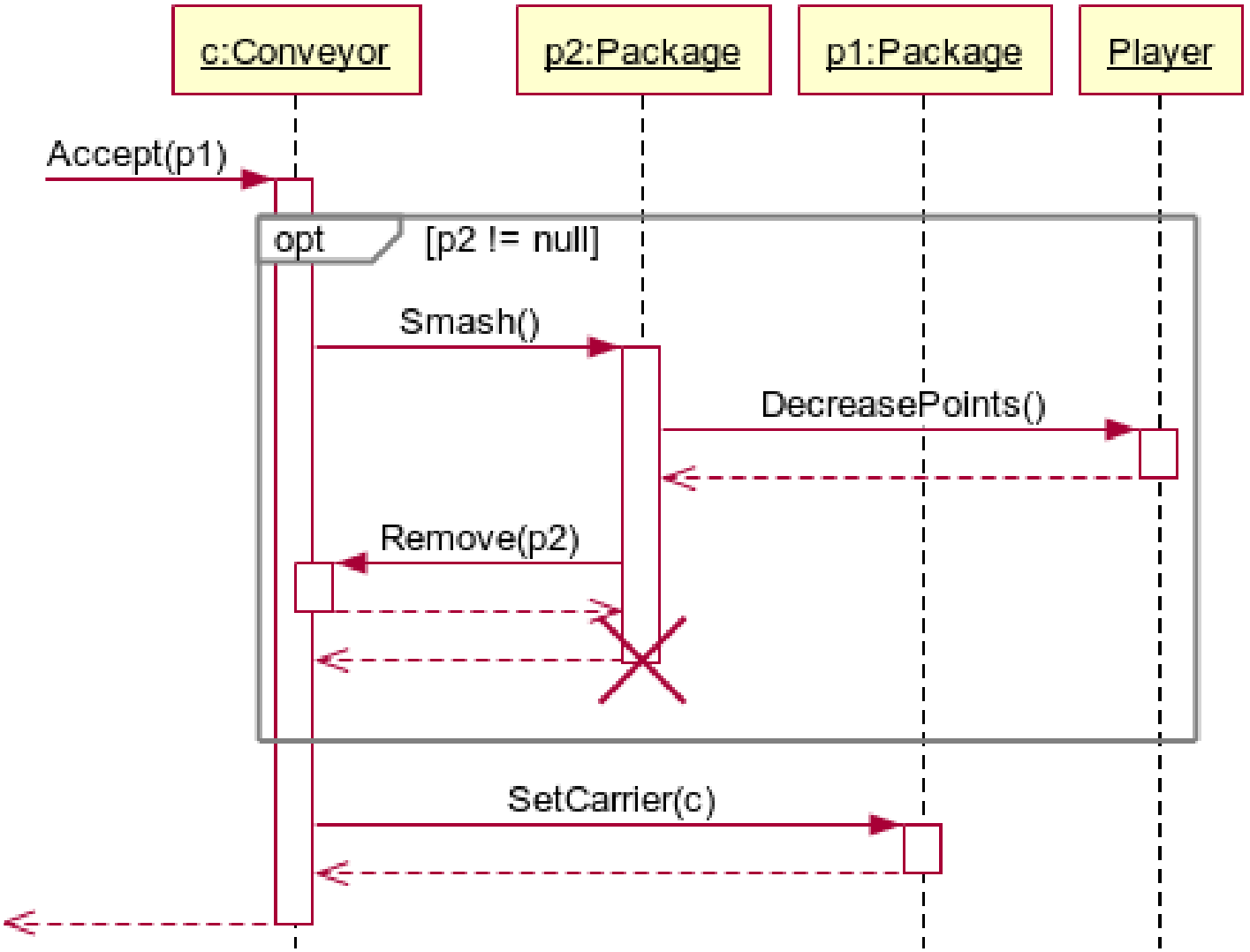
Perishable package steps



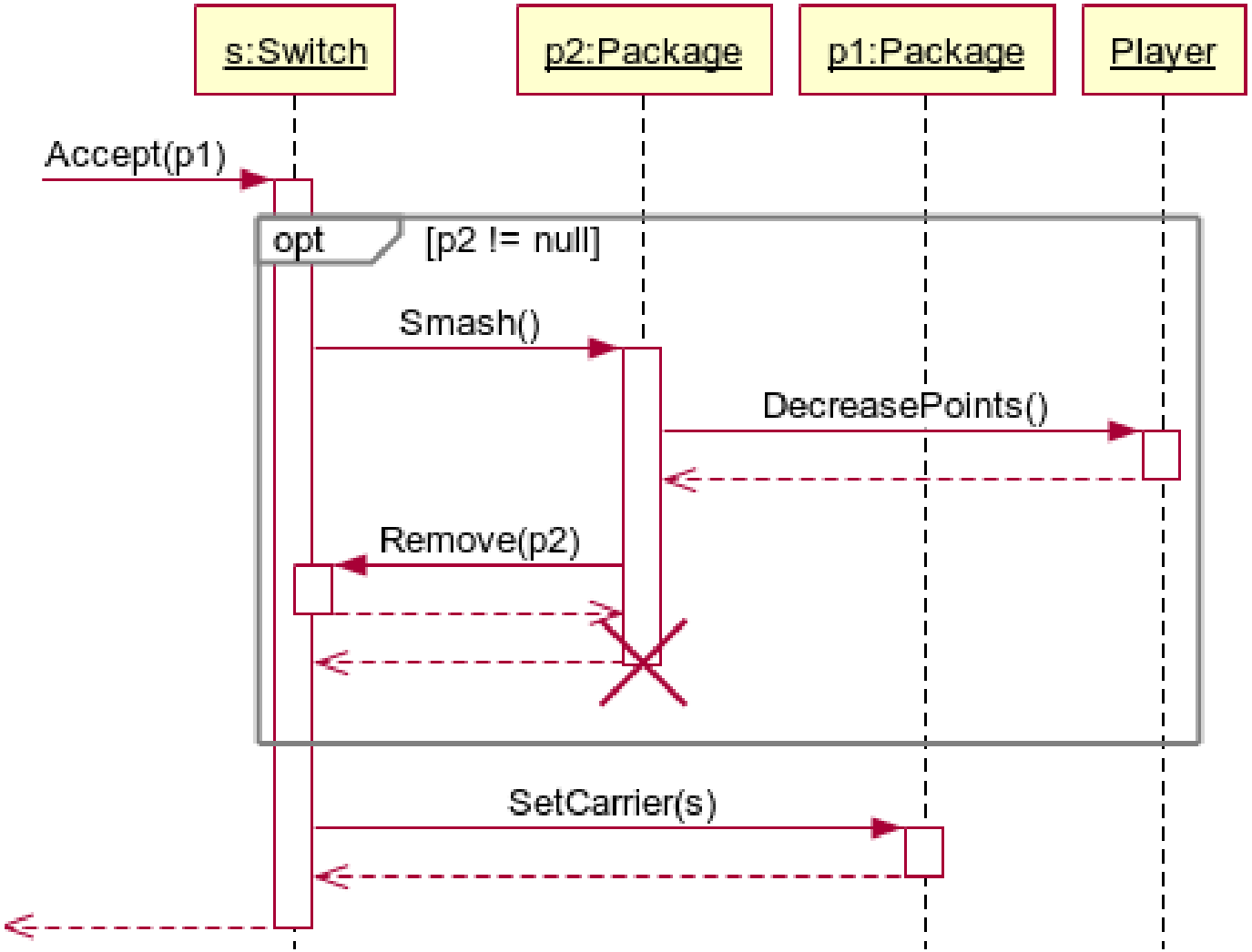
Change direction of the switch



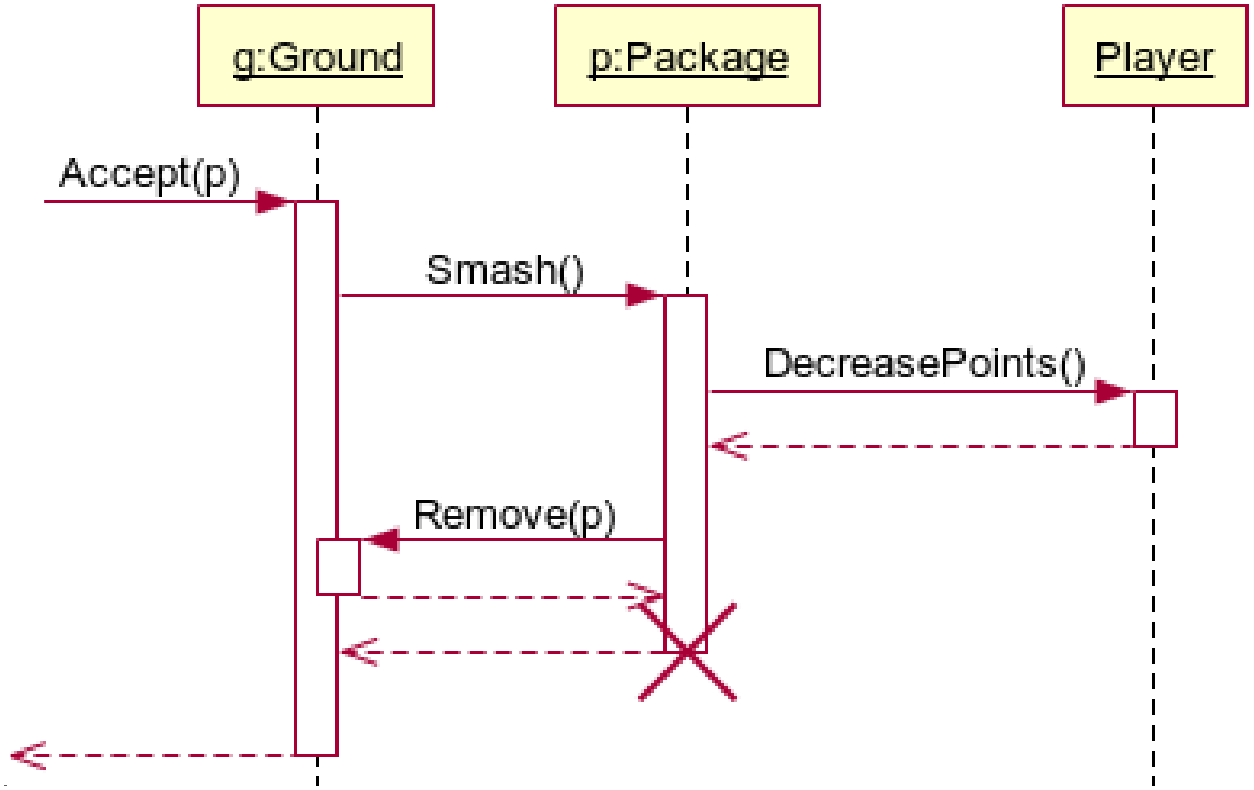
Package to conveyor



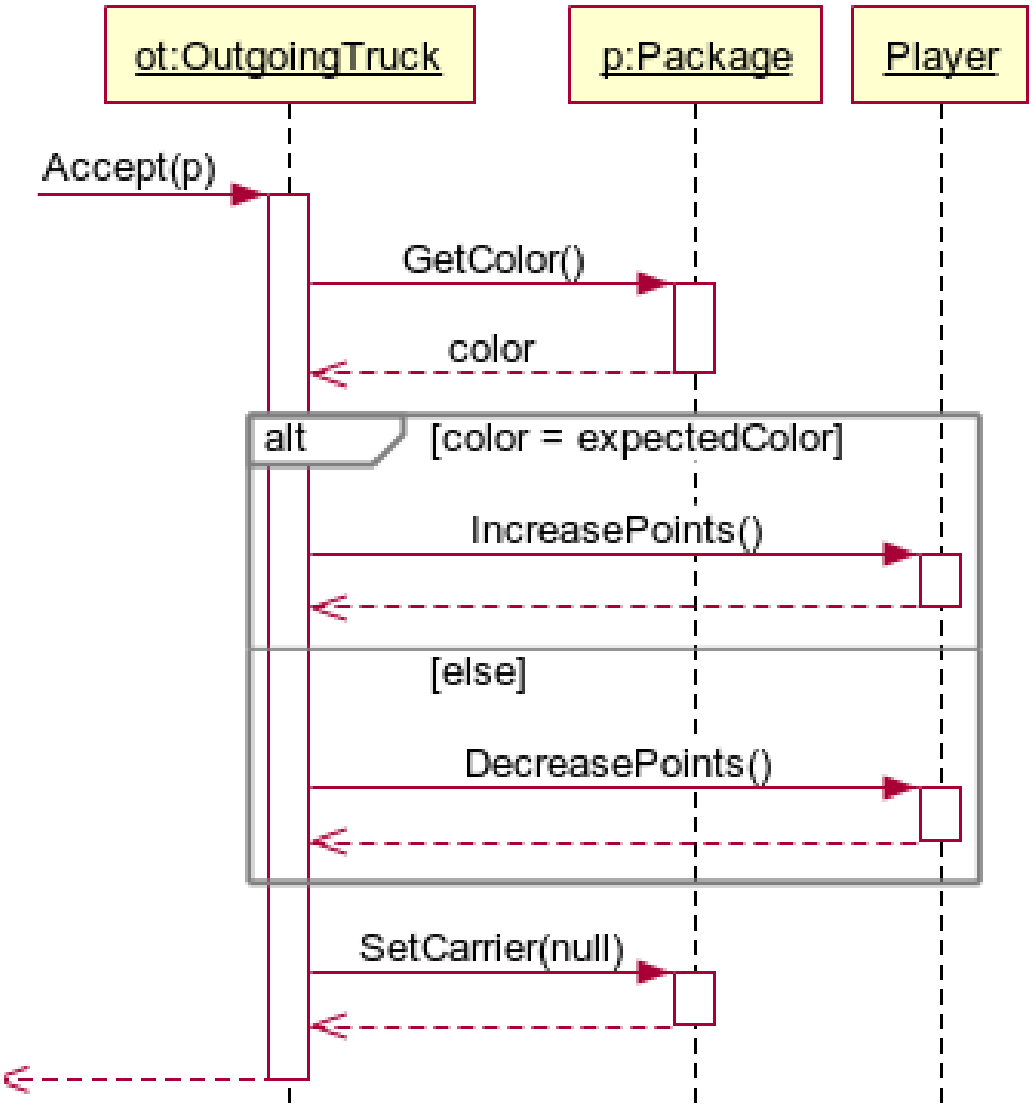
Package to switch



Package to ground



Package to outgoing truck



Alternative designs

Alternative designs (all of these are accepted)

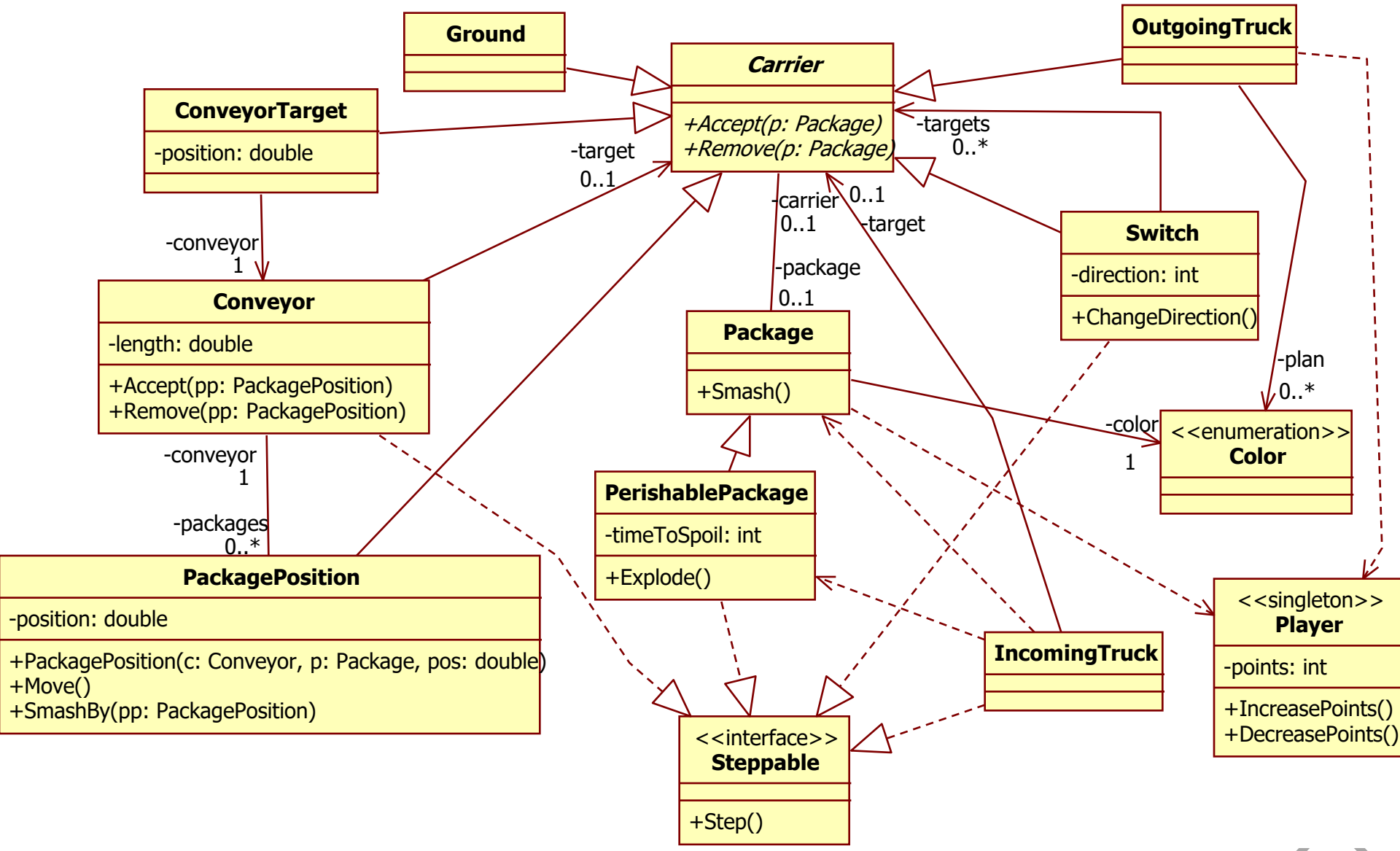
- Switch is a descendant of the Conveyor
  - since most sequences are the same
  - Switch does not store direction as an index, instead, the ancestor Conveyor's target carrier is used to define the direction
  - (Conveyor as a special case, i.e., as a descendant of the Switch is incorrect: Switch has more responsibilities)
- Fields instead of a general graph
  - but without parallel layers: the conveyor/switch/ground *is* a field, *not on* the field
  - *no HitBy callbacks* as in the Pacman: there is no need to find out the types of the packages
- A conveyor with length and multiple packages
  - position is required, but only on the conveyor: wrapper for the package
  - Conveyor is not the Carrier, the wrapper is the Carrier
- A conveyor of multiple pieces, max. 1 package per piece
  - no position is required
  - Conveyor is not the Carrier, Piece is the Carrier
- A switch targets only conveyors
- Trucks connect to entry/exit points
  - trucks are not directly connected to conveyors/switches/etc.
- PerishablePackage does not need to be separated, can be merged with Package
- etc.

Other design questions

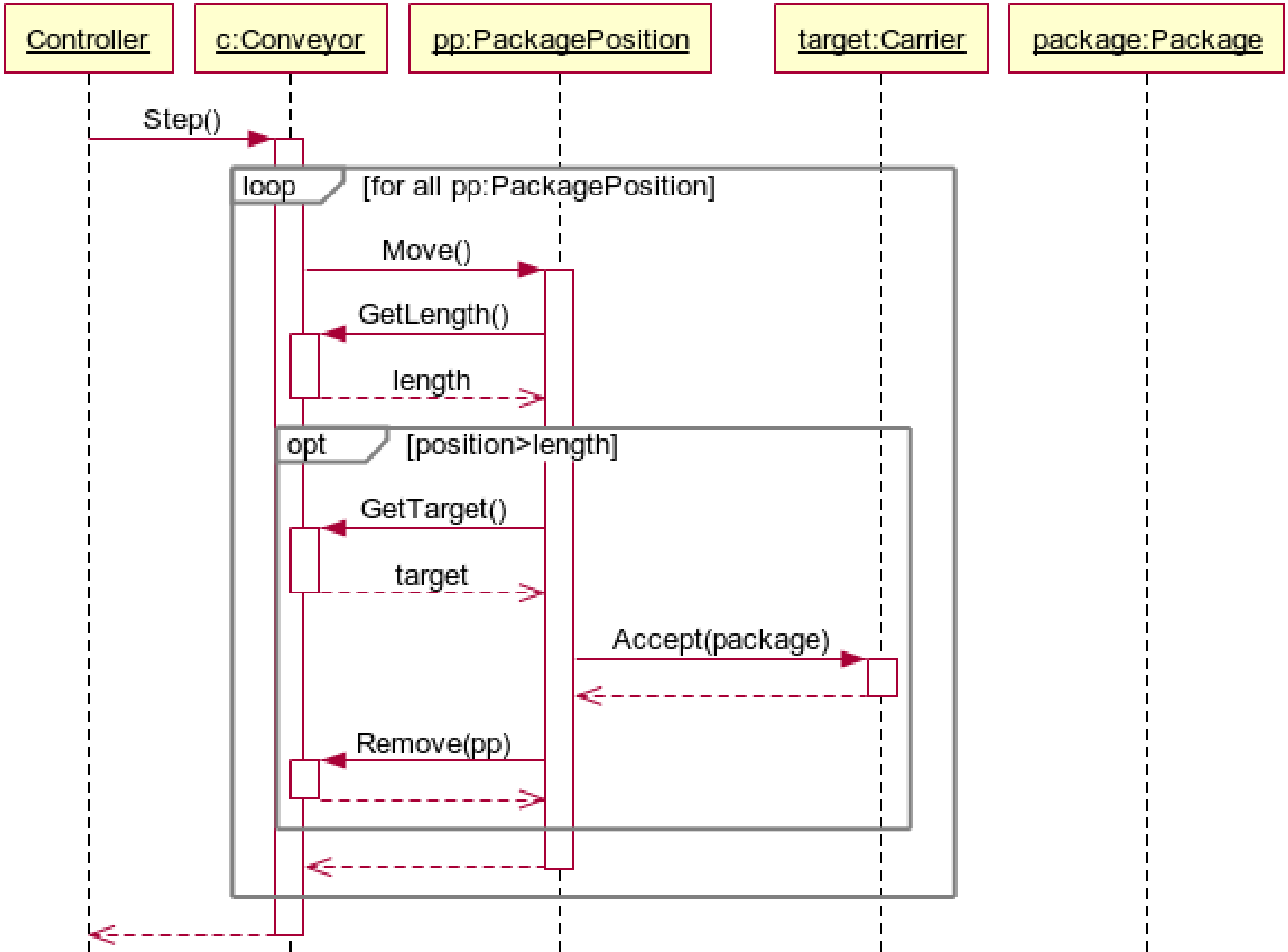
- What if the trucks store the packages?
  - Package needs more methods: LeaveTruck(), ArriveToTruck()
    - these can help to start and stop the spoiling
  - IncomingTruck: does not create packages, just forwards them to the target carrier, calls LeaveTruck()
  - OutgoingTruck: sets the package's carrier to be the truck, calls ArriveToTruck()
- What if the same truck can be incoming or outgoing depending on its place?
  - only one Truck class, it is a Carrier, functions of the IncomingTruck and OutgoingTruck are merged

Alternative design: a conveyor with length

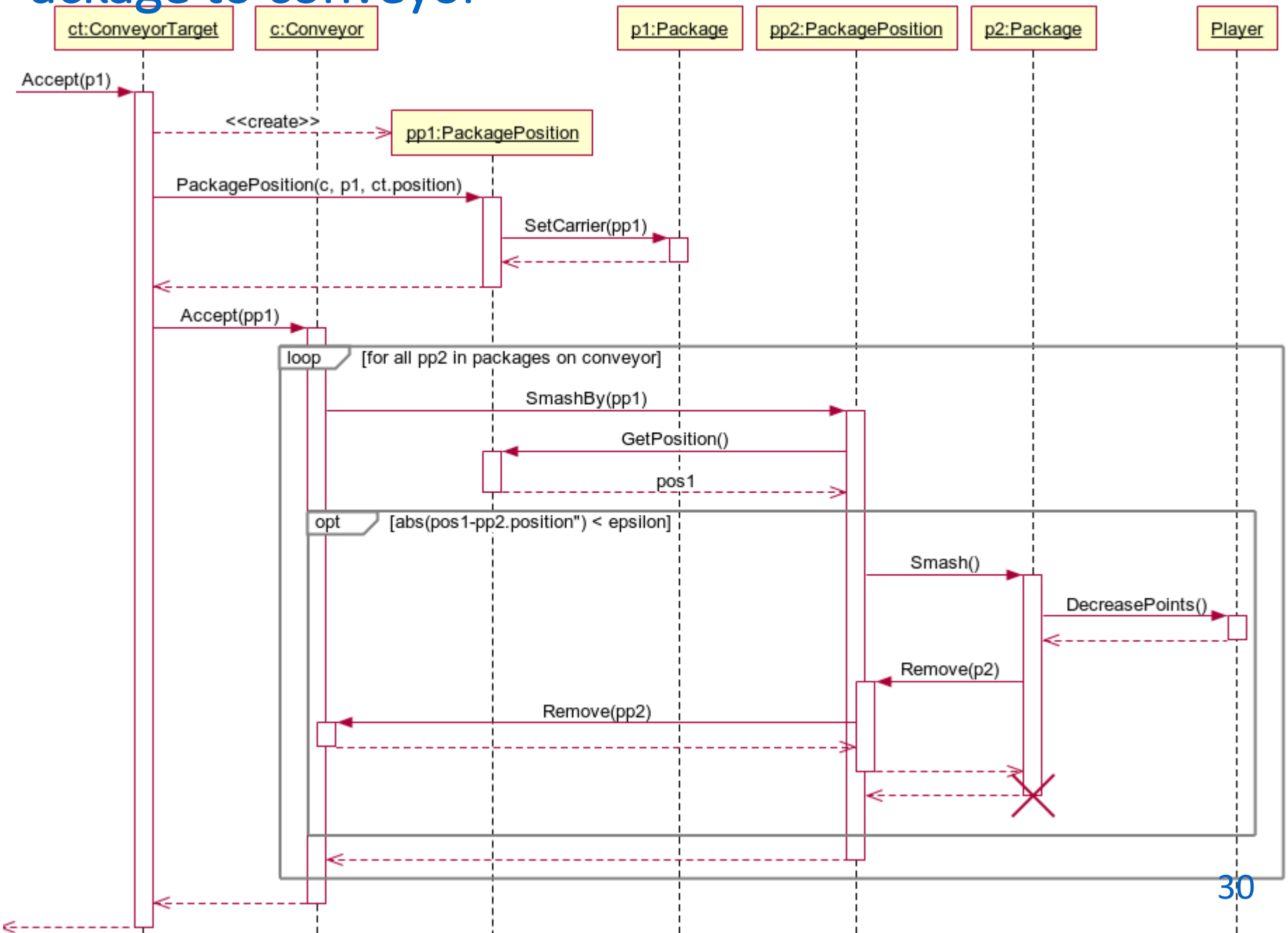
Conveyor with length and multiple packages



Conveyor steps



Package to conveyor



Other sequence diagrams

- Same as before:
  - Making elements step
  - Switch steps
  - Incoming truck steps
  - Perishable package steps
  - Change direction of the switch
  - Package to ground
  - Package to switch
  - Package to outgoing truck