SOLID Exercise: The portioning machine

Introduction to the exercise

This exercise is designed to train you in several things when it comes to software design: To understand a new problem domain quick, to create a domain model, and – most importantly – to create an extensible, simple design to meet specified requirements. To create such a design, you must adhere to the SOLID principles.

Note: The description of the individual exercises is loose by design: *You* must evaluate the alternatives and make the important decisions.

Introduction to the problem domain:

A *portioning machine* makes portions of a predefined weight out of items, e.g. 2000-gram portions of chicken breast. The portions are put together so that they get as close to, but not under, the target weight as possible.

The machine, as sketched in Figure 1 below, consists of three sections of conveyor bands: an *in-feed*, a *weight*, and a *portioner*. The in-feed accelerates the items to the speed of the remaining sections. The weight weighs the items traveling over it, and the portioner ejects the items into *bins*. When a bin is full, i.e. at or over target weight, it is emptied into a container below it.

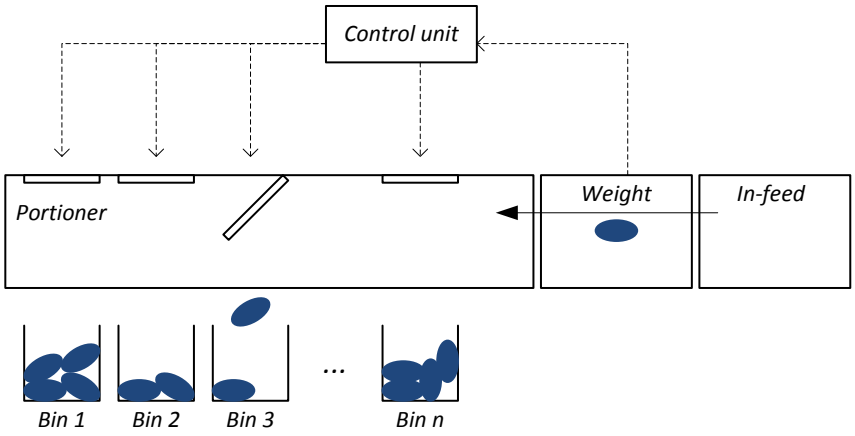

Figure 1: Portioner machine

Naturally, to be competitive, the portioner machine must ensure two things:

1. The bins are filled as little over the target weight as possible. The "overweight" is termed "give-away"[i].
2. No item remains unassigned, e.g. travels over the edge of the portioner and is dropped on the floor. It is better to use an item to overfill a batch than it is to lose the item.

The portioner machine is controlled by a *control unit* which receives a signal when an item is on the weight, calculates to which bin the item must be assigned, and signals the corresponding ejector to eject the item at the proper time. The core functional requirements for the control unit are as follows:

REQ1: When an item arrives, the arrival must be logged
REQ2: When an item arrives, it must be assigned to a bin.
REQ3: When a bin is at or over its target weight, it must be emptied.
REQ4: When a bin is emptied, the weight, target weight and "give-away" in percent must be logged.

In this exercise, you will design and implement the software for the portioner machine's control unit to meet the above requirements

Exercise 1:
Sketch a domain model of the system to ensure that you understand the conceptual classes in the problem domain and the collaborations between them. The domain model must be of a system which fulfills the above requirements.

Exercise 2:
Create a design for the system that satisfies the above requirements using a simple Round-Robin algorithm to assign items to bins (item 1 goes in bin 1, item 2 goes in bin 2, etc.). Of course, your design should be extensible and "clean" Things you may want to consider extension points are the number of bins, their target weights, the algorithm used to assign items to bins, logging media, etc.

Consider the weight external to the system you are designing. An implementation of a general "item provider" which will yield items of a normal distribution is available from BB
.

Exercise 3:
Implement your design from 2. Note how the "give-away" using the Round-Robin algorithm is fairly high.

Exercise 4:
Another less simple algorithm for item assignment is to let the bins "score" the items: The bins are each asked to score, e.g. on a scale 0.0-1.0, their desire to have a given item. The item is then assigned to the bin which yields the highest score. The score could e.g. be dependent on how close the item would bring the bin to be filled "perfectly" etc. Design, implement and use such a portioning algorithm in which the bins score the items. If you need inspiration on the algorithm, contact your teacher.

[i] Note to geeks: Some *very* clever algorithms exist to minimize give-away above. For example, check out the Maximum Coverage problem.