Design Decisions and Assumptions:

Firstly I assumed that the printers are being used by a commercial/office place rather than being produced at a manufacturer. This assumption is an important one since I picked my data structure, array, based on this assumption. Since an office space has a fixed amount of printers we can assume a fixed size for the array data structure. This would give us an edge as we'd have constant look up on any given printer registered at the office. If the opposite were the case, meaning if the printers were being produced rather than just being used. I would've used a linked list since this would give me an "infinite" scaling for the number of printers I could have, theoretically.

In terms of implementation, I decided that I'd do a dynamic allocation for the array as not every office would have the same number of printers. This allows for some flexibility for the Printer class. My other decision was to use a bool array. Where false represents a free printer and true represents a busy printer. In the constructor, I initialized all the printers to false indicating that it'd be free when you instantiate it. I also kept a variable tracking the free printers. That way I don't have to loop and count each time someone wants to know how many free printers we have. Furthermore, since it's a binary state of being free and busy we can know how many busy printers we have by doing the number of printers minus free printers. Each time we use a printer we decrement the free printer variable by one and vice versa for freeing the printer.

Resources

http://www.cplusplus.com/reference/queue/queue/

Peers -
Kcirde, Kassandra, Garrett