REFLECTION DOCUMENT

Why did I choose to implement the problem through Strategy Design Pattern?

I chose this pattern because the class behaviour or its algorithm can be changed at run time. In this, we create objects which represent various strategies and a context object whose behaviour varies as per its strategy object. The strategy object changes the executing algorithm of the context object. In this problem we have two algorithms in which the jobqueue can be stored. The operations that are to be followed with two different algorithms are addToQ (String id), removeFromQ(), isEmpty(), size() that changes its behaviour with respect to the type of implementation user chooses and asks Context Class to use.
Strategy pattern uses composition instead of inheritance allowing better decoupling between the behaviour and the class that uses the behaviour.

JobQueue- is the context class which refers to Strategy Interface for performing an Algorithm, making context class independent of how an algorithm is implemented.

Queue- This is an **interface,** also called **Strategy**, where we decide exactly which strategy should be used.

ArrayQ and LinkedQ – are the **ConcereteStrategy** classes that implement the interface Queue, implements respective algorithms.
addToQ (String id), removeFromQ(), isEmpty(), size() – operations that define the behaviour of the ConcreteStrategy to be used.

Client- has the static main() which refers to the Context.