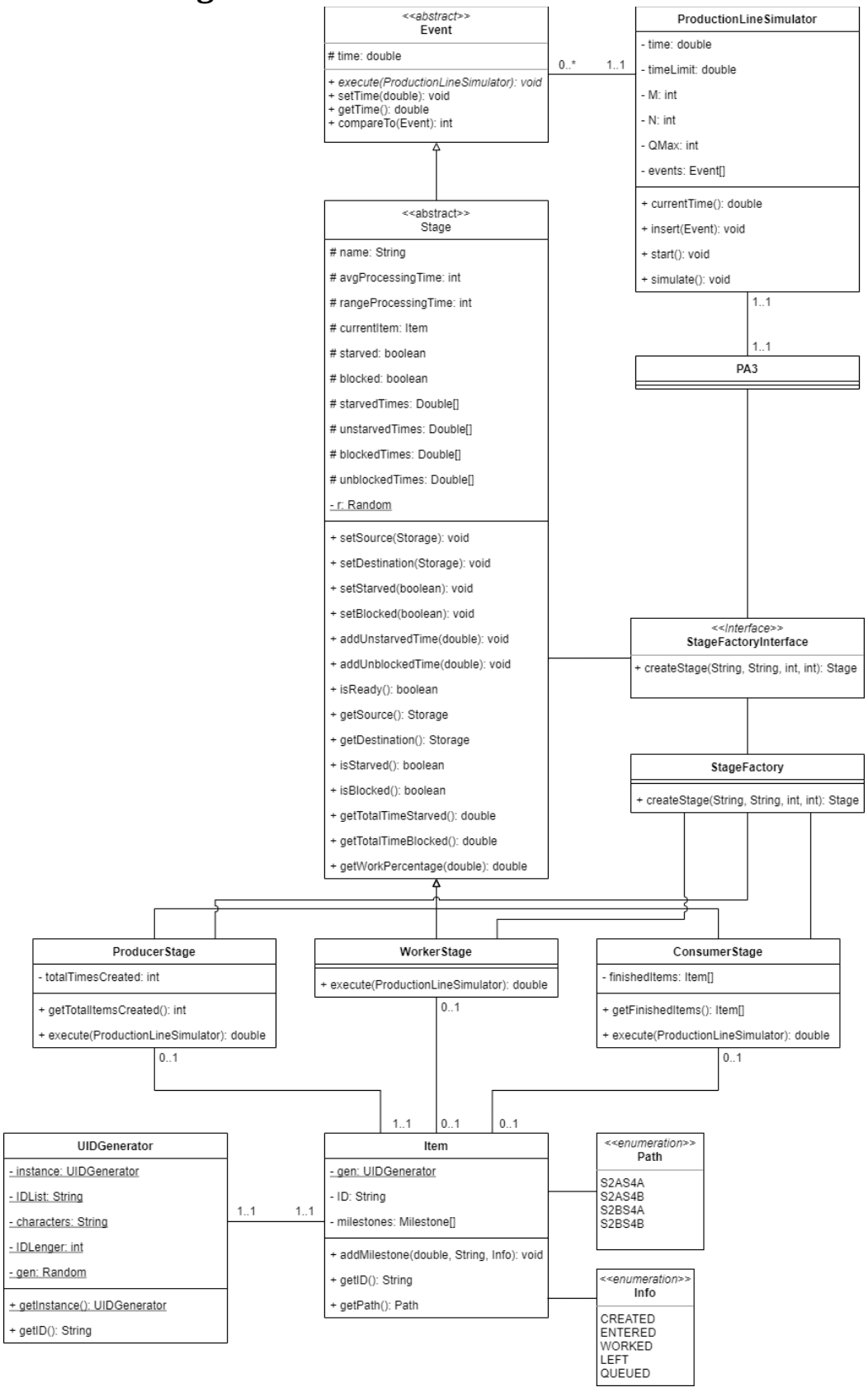
Assignment 3 Report

1 UML Class Diagram



2 Use of Inheritance and Polymorphism

In my program, to implement discrete event simulation, I decided that the production stages would be able to create an event that can be executed. The base abstract class Event details what it takes to be executed by the ProductionLineSimulator. This is extended by the abstract class Stage holds the shared functionality and attributes between all the stages. The ProducerStage, WorkerStage and ConsumerStage all extend the base Stage. They contain their class specific execute methods and some unique attributes. This is the only use of inheritance and polymorphism in my program. I also used a factory pattern to create the stages in the ProductionLineSimulator.

3 Catering for Different Topology

It would be very easy to create or remove more stages in my program as all it takes is making a new stage and associated storage and setting up the new relationships between the stages in the ProductionLineSimulator::start() method. To set up tri-parallel stages it would just be updating the destinations and sources of the inter-stage storages and setting up the relationship for the new stage.

4 Complex Production Line

To alter the production line for a more complex scenario, such as taking two different types of items to make a new one, change would have to be made to how items are defined. This could easily be done by making an abstract Item superclass which subclasses could then inherit from. Production stages would hold the abstract Item class which could allow for different types of items to be in the production stages. For new types of production stages they could inherit from the Stage abstract class if they share functionality with that class. The Stage abstract class would have to be changed to allow for multiple sources or destinations. I would design my program to be more extensible and configurable. I would have made classes such as the Item or even Storage to be abstract first and then implemented so different versions could be made.