SENG 300 GA Iteration 3

Dany Saaid Bonilla - 30050992
Grace Heemeryck - 30068546
Matt Jarrams - 30061191
Nathan Lum - 30064695
Mathew Luong - 30068650
Zeyad Omran - 30096692
Kylie Sicat - 30062029
Mackenzie Dalton- 30063061
Joseph Lam - 30065199
Kirk Elumir - 30073334
Sarthak Sharan - 30092835

Explanation:

Structural Diagrams:
- The class SelfCheckoutSoftware contains methods/member variables with all the ListenerStubs, Databases and Object classes. This is denoted with the aggregation arrows.
- ListenerStubs: all implement their respective Listeners (from Hardware) this is denoted with the implementation dotted arrows.
- All the Listeners extend AbstractDeviceListener (from Hardware) this is denoted with the inheritance arrows.

GUI Structural Diagram
- All the GUI classes contain methods/member variables (i.e the control variable). This is denoted with the aggregation arrows.
- All the GUI Panels extend Jpanel (from Swing GUI classes) this is denoted with the inheritance arrows.

Sequence Diagrams
- In the scanItem and PlaceItemInBaggingArea diagrams, a BarcodedItem object is created/instantiated, this is denoted by the <<create>> arrow
- Roles are used to model generic objects, e.g. SelfCheckoutSoftware is a role in the diagrams, roles that have no prefix before the : are anonymous (s:SelfCheckoutSoftware means s is the name of an instance of SelfCheckoutSoftware)
- The role of type map<Barcode, BarcodedProduct> is the productdatabase, and map<Product, Integer> is the inventory database (these are fields in the software class)
- Alt frames are used frequently to model if/else constructs, where the if condition is contained within a guard, denoted by square brackets e.g. [condition]
- In PayWithCard, ScanItem, and PlaceItemInBaggingArea, return statements connected to the SelfCheckoutSoftware lifeline denote the method returning true/false

State Diagrams
- The state diagrams try to show the most important parts of our system and the states associated with those processes. We have a high level one to show how the whole system reacts in different states and how it gets to them. But we also make sure to show some of the lower level processes since they have their own states they go through that impact the higher level of the system.
- This is shown through a checking items, bagging area and a paying state diagram (the major functionalities, we thought it was best to abstract other smaller state diagrams)

Although we do understand our design could have been better (we could have separated some more classes to avoid high coupling) at the time we thought this was the best way to proceed at the time especially with large amounts of changes and not knowing the full scope of the project beforehand. This is what made our design good as it was easy to adapt to changes. However, due to the high coupling these changes did create other problems in the system as well.