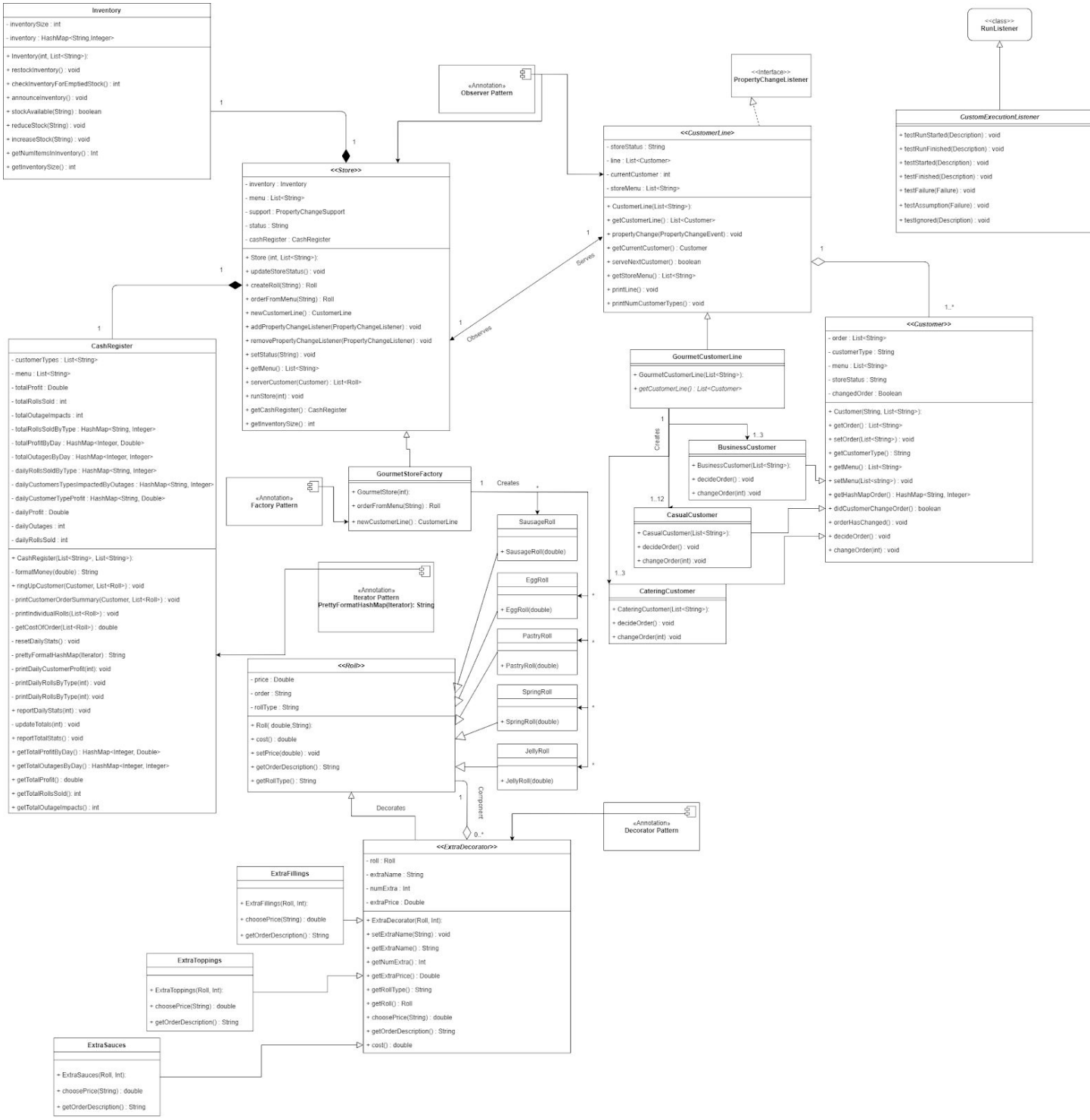
Inventory
- inventorySize : int
- inventory : HashMap<String,Integer>
+ Inventory(int, List<String>):
+ restockInventory() : void
+ checkInventoryForEmptiedStock() : int
+ announceInventory() : void
+ stockAvailable(String) : boolean
+ reduceStock(String) : void
+ increaseStock(String) : void
+ getNumItemsInInventory() : Int
+ getInventorySize() : int
«Annotation»
Observer Pattern
<<Interface>>
PropertyChangeListener
<<class>>
RunListener
CustomExecutionListener
+ testRunStarted(Description) : void
+ testRunFinished(Description) : void
+ testStarted(Description) : void
+ testFinished(Description) : void
+ testFailure(Failure) : void
+ testAssumption(Failure) : void
+ testIgnored(Description) : void
<<Store>>
- inventory : Inventory
- menu : List<String>
- support : PropertyChangeSupport
- status : String
- cashRegister : CashRegister
+ Store (int, List<String>):
+ updateStoreStatus() : void
+ createRoll(String) : Roll
+ orderFromMenu(String) : Roll
+ newCustomerLine() : CustomerLine
+ addPropertyChangeListener(PropertyChangeListener) : void
+ removePropertyChangeListener(PropertyChangeListener) : void
+ setStatus(String) : void
+ getMenu() : List<String>
+ serverCustomer(Customer) : List<Roll>
+ runStore(int) : void
+ getCashRegister() : CashRegister
+ getInventorySize() : int
<<CustomerLine>>
- storeStatus : String
- line : List<Customer>
- currentCustomer : int
- storeMenu : List<String>
+ CustomerLine(List<String>):
+ getCustomerLine() : List<Customer>
+ propertyChange(PropertyChangeEvent) : void
+ getCurrentCustomer() : Customer
+ serveNextCustomer() : boolean
+ getStoreMenu() : List<String>
+ printLine() : void
+ printNumCustomerTypes() : void
Serves
Observes
<<Customer>>
- order : List<String>
- customerType : String
- menu : List<String>
- storeStatus : String
- changedOrder : Boolean
+ Customer(String, List<String>):
+ getOrder() : List<String>
+ setOrder(List<String>) : void
+ getCustomerType() : String
+ getMenu() : List<String>
+ setMenu(List<string>) : void
+ getHashMapOrder() : HashMap<String, Integer>
+ didCustomerChangeOrder() : boolean
+ orderHasChanged() : void
+ decideOrder() : void
+ changeOrder(int) : void
CashRegister
- customerTypes : List<String>
- menu : List<String>
- totalProfit : Double
- totalRollsSold : int
- totalOutageImpacts : int
- totalRollsSoldByType : HashMap<String, Integer>
- totalProfitByDay : HashMap<Integer, Double>
- totalOutagesByDay : HashMap<Integer, Integer>
- dailyRollsSoldByType : HashMap<String, Integer>
- dailyCustomersTypesImpactedByOutages : HashMap<String, Integer>
- dailyCustomerTypeProfit : HashMap<String, Double>
- dailyProfit : Double
- dailyOutages : int
- dailyRollsSold : int
+ CashRegister(List<String>, List<String>):
- formatMoney(double) : String
+ ringUpCustomer(Customer, List<Roll>) : void
- printCustomerOrderSummary(Customer, List<Roll>) : void
- printIndividualRolls(List<Roll>) : void
- getCostOfOrder(List<Roll>) : double
- resetDailyStats() : void
- prettyFormatHashMap(Iterator) : String
- printDailyCustomerProfit(int): void
- printDailyRollsByType(int) : void
- printDailyRollsByType(int): void
+ reportDailyStats(int) : void
- updateTotals(int) : void
+ reportTotalStats() : void
+ getTotalProfitByDay() : HashMap<Integer, Double>
+ getTotalOutagesByDay() : HashMap<Integer, Integer>
+ getTotalProfit() : double
+ getTotalRollsSold(): int
+ getTotalOutageImpacts() : int
GourmetCustomerLine
+ GourmetCustomerLine(List<String>):
+ getCustomerLine() : List<Customer>
Creates
BusinessCustomer
+ BusinessCustomer(List<String>):
+ decideOrder() : void
+ changeOrder(int) :void
CasualCustomer
+ CasualCustomer(List<String>):
+ decideOrder() : void
+ changeOrder(int) :void
CateringCustomer
+ CateringCustomer(List<String>):
+ decideOrder() : void
+ changeOrder(int) :void
«Annotation»
Factory Pattern
GourmetStoreFactory
+ GourmetStore(int):
+ orderFromMenu(String) : Roll
+ newCustomerLine() : CustomerLine
Creates
SausageRoll
+ SausageRoll(double)
EggRoll
+ EggRoll(double)
PastryRoll
+ PastryRoll(double)
SpringRoll
+ SpringRoll(double)
JellyRoll
+ JellyRoll(double)
«Annotation»
Iterator Pattern
PrettyFormatHashMap(Iterator): String
<<Roll>>
- price : Double
- order : String
- rollType : String
+ Roll(double,String):
+ cost() : double
+ setPrice(double) : void
+ getOrderDescription() : String
+ getRollType() : String
Decorates
Component
«Annotation»
Decorator Pattern
<<ExtraDecorator>>
- roll : Roll
- extraName : String
- numExtra : Int
- extraPrice : Double
+ ExtraDecorator(Roll, Int):
+ setExtraName(String) : void
+ getExtraName() : String
+ getNumExtra() : Int
+ getExtraPrice() : Double
+ getRollType() : String
+ getRoll() : Roll
+ choosePrice(String) : double
+ getOrderDescription() : String
+ cost() : double
ExtraFillings
+ ExtraFillings(Roll, Int):
+ choosePrice(String) : double
+ getOrderDescription() : String
ExtraToppings
+ ExtraToppings(Roll, Int):
+ choosePrice(String) : double
+ getOrderDescription() : String
ExtraSauces
+ ExtraSauces(Roll, Int):
+ choosePrice(String) : double
+ getOrderDescription() : String