# **Model**

*Package in package ''*

Model

Version Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

## **Class Structure**

*Package in package 'Model'*

Class Structure

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

### **Execution and tests class structure diagram**

*Class diagram in package 'Class Structure'*

Execution and tests class structure

Version 1.0

lusip created on 6/14/2020. Last modified 6/14/2020



Execution and tests class structure

### **Note**

*Note in package 'Class Structure'*

The MainTest class is a Junit kind of class that tests the whole app. It calls the main method as if it came from the command line and then it adds AssemblyLines to the AssemblyLineServer, which is a kind of "listener" for new assemblylines.

In this scheme the assembly lines are autogenerators of food products.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

Extends

### **Main**

*Class in package 'java'*

A simulation of the following use case: "We have to provide a solution for a factory that processes food products in different assembly lines. We need to automate the cooking portion of the assembly line for the products. The factory produces various food products, which are created in assembly lines. There is a common part to many of them, and that is the cooking stages for which some ovens are used. The process to cook the different products (in the â€œcooking stageâ€) in a given point of the assembly line involves getting the products from the line, to put them in the oven for a specific amount of time, in the order they arrive. An intermediate store is used for the products that arrive, if there is no space left in the oven, which has a finite size. If there is no more room in the ovens or the stores when extracting it from the assembly line, the originating assembly line halts. After each product is cooked, we have to extract it and return it to the originating line (this is because multiple lines arrive at this automated stage). We have to develop an application that controls the cooking stage of the factory."

Main

Version 1.0 Phase 1.0 Proposed

hernan ezequiel martinez created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: Main : Class, Public  To: KitchenBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: Main : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLineServer : AssemblyLineServer Private = null  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| getAssemblyLineServer () : AssemblyLineServer Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| main (args : String[] ) : void Public  This whole method, just builds the kitchen and starts to "hear" for requests. This stays looping. If this were an "actual" app, I'll put this hearing requests in a socket. Instead of it, we will work with a Junit that will hit some shared objects and notify it of a new assembly line.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| startAseemblyLineServer () : void Private  Starts a looping thread that heards for request from different assembly lines.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **AssemblyLineServer**

*Class in package 'view'*

Receives all the requests from an external source. For this implementation, we will use a unit test. For future use, we will go for the implementation of a REST api (Throught Spring MVC) or hearing in a socket directly. Its major aim is to build the kitchen, start it up. And hear for new assembly lines to appear, dynamically over an external source. Every need assembly line will generate its own products.

AssemblyLineServer

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: AssemblyLineServer : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private Const  This is not how the request will be implemented in a real app!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| endProgram : boolean Private = false  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| kitchen : Kitchen Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| startDateTime : LocalDateTime Private  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private kitchen (Class) Kitchen |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| addAssemblyLine () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| AssemblyLineServer (kitchen : Kitchen ) : Public  Defines the major controllers to operate the simulation. It starts the kitchen and defines the assembly lines.  Properties:  throws = KitchenRequiredException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getActiveSecondsSinceStart () : Long Public  Just returns the amount of seconds since the current object started to hear requests.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getAssemblyLines () : List<AssemblyLine> Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getKitchen () : Kitchen Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isEndProgram () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  We kill the process in cold blood; losing state and data.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printFinishedProductsInOrder () : void Public  This prints in the standard output the products already processed and finished. We use this only in the context of this simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatusAllAssemblyLines () : void Public  We go AssemblyLine by AssemblyLine Asking for the number of Elements in its "waiting" and "finished" lines.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setEndProgram (endProgram : boolean ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| waitForNewAssemblyLines () : void Public  Loops for requests over the creation of new assembly lines (Threads), and starts all the simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **MainTest**

*Class in package 'java'*

Executes and test different the creation of new assembly lines and puts the simulation in motion.

MainTest

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: AssemblyLineServer : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: Main : Class, Public |

| **OPERATIONS** |
| --- |
| mainAppStartUpAndStopTest () : void Package  Properties:  annotations = @Test  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| runAssemblyLinesProductGeneratorTest () : void Package  Properties:  annotations = @Test  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **Kitchen Class Structure diagram**

*Class diagram in package 'Class Structure'*

Kitchen Class Structure

Version 1.0

lusip created on 6/14/2020. Last modified 6/14/2020



Kitchen Class Structure

### **Note**

*Note in package 'Class Structure'*

The AssemblyLine is a collection of objects that is representative of the state of all the threads working; this whole object has to be implemented in a thread-safe kind of structure because is going to be accessed by several threads at once.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

Extends

### **Cooker**

*Class in package 'controllers'*

Handles the thread of execution of the Kitchen. What needs to be cooked, in which order. It workd, more or less, like a "business controller" for the backend.

Cooker

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private = null  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovens : List<Oven> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stores : List<Store> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| timers : List<Executor> Private Const = new CopyOnWriteArrayList<>()  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private cooker (Class) Cooker |

| **OPERATIONS** |
| --- |
| addNextFinishedProductToAssemblyLine (idAssemblyLine : Integer , product : Product ) : void Private  Adds one more product to the cache, for its given AssemblyLine  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| cook (product : Product ) : Boolean Public Const  Here lies the main logic of how to put to cook a given product  Properties:  throws = InterruptedException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Cooker (ovens : List<Oven> , stores : List<Store> ) : Public  This constructor allows us to maintain a reference to the Kitchens ovens & stores for the cooker to manage them.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductInOven (product : Product ) : Boolean Private  Tries and places a product in an Oven, if it is successful it starts a timer that simulates the cooking time.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductInStorage (product : Product ) : Boolean Private  Tries to place the product in a Storage, if not, it returns false.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLines (assemblyLines : List<AssemblyLine> ) : void Public  We assign the assemblies lines to have access to the finishedProducts sorted queue.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| startCookingTimer (product : Product ) : void Private  Start a thread with a timer to hold on it; when it finishes, we take the product out of the oven.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOffAllOvens () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

|  |
| --- |
| turnOnAllOvens () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

### **Kitchen**

*Class in package 'controllers'*

This is the kitchen, it is composed of many ovens and stores. Assumption: the stores aren't indexed as one per OvenImpl; and the ovens doesn't has stores associated. Number of Ovens: N and Stores: M The kitchen starts the 'cooker' who is a virtual representation of a guy that performs the task of cooking and handling the products to the due assembly lines. There is a requirement: the products should be placed in the line, again in the same order in which they appeared. We solved this with an implementation of a "binary tree" that java has implemented as a "PriorityQueue"; this sort of data structure has the peculiarity of receiving items in any order and then retrieving them ordered. Neat!

Kitchen

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| cooker : Cooker Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| endKitchen : boolean Private = false  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executorCooker : ExecutorService Private Const = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executorDispatcher : ExecutorService Private Const = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| lineContinues : AtomicBoolean Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovens : List<Oven> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stores : List<Store> Private Const  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private cooker (Class) Cooker |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private kitchen (Class) Kitchen |

| **OPERATIONS** |
| --- |
| addOven (oven : Oven ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| addStore (store : Store ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  Brutal! but necessary option...  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Kitchen () : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductOrHalt (assemblyLine : AssemblyLine , product : Product ) : void Private  It searchs a place for the taken product, if there isn't any...the line halts.  Properties:  throws = InterruptedException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLines (assemblyLines : List<AssemblyLine> ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| start () : void Public  Starts the kitchen's "Cooker". A thread that monitors the assembly lines to take products to cook and retrieve them.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| stop () : void Public  Procedurally STOPS the whole set of processes as is.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| tryGettingProductFromStoresFirst () : Product Private  Before we go for the AssemblyLine, we go into the Stores and checkout for any remaining item to cook!  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **AssemblyLine**

*Class in package 'model'*

Represents the implementation of an hypothetical "Food Products Assembly Line" that is about to be simulated.

AssemblyLine

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from AssemblyLine to AssemblyLineStage  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| continueLine : Boolean Private = true  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executor : ExecutorService Private = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| FINISHED\_PRODUCTS\_INITIAL\_CAPACITY : int Private Const = 11  A prime number for good luck. :)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| finishedProducts : PriorityBlockingQueue<Product> Private Const = new PriorityBlockingQueue<>(FINISHED\_PRODUCTS\_INITIAL\_CAPACITY,orderNumberSorter)  OUT queue  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| id : Integer Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_COOK : int Private Const = 15  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_SIZE : int Private Const = 30  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCTIVITY\_DELAY : int Private Const = 3  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_COOK : int Private Const = 5  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_SIZE : int Private Const = 10  DISCLAIMER: These values do not represent REAL cooking times neither food product sizes (just intented for this simulation use).  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCTIVITY\_DELAY : int Private Const = 0  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| orderNumberSorter : Comparator<Product> Package Const = Comparator.comparing(product -> ((Food)product).getOrderNumber())  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| production : AtomicInteger Private = new AtomicInteger(0)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| PRODUCTION\_TIME : long Private Const = 3  I like it every three seconds, a prime number takes the oddities to see some sunshine!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stopProduction : AtomicBoolean Private = new AtomicBoolean(false)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| waitingProducts : ConcurrentLinkedQueue<Product> Private Const = new ConcurrentLinkedQueue<>()  IN queue  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| addProduct (product : Product ) : void Private  Adds the product to the "line" of products that need to be cooked by the Cooker on the multiple ovens.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

|  |
| --- |
| AssemblyLine (id : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| continueProduction () : void Public  Wakes up the production.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| generateRandomProduct () : Product Private  I randomize the values that a new product might have, just before it enters the "input" line. I tries with minimal and maximum values but, ideally, those must be out of some sort of "setup scheme".  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getId () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| haltProduction () : void Public  Stops the production, but the thread continues.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isHalted () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printAllFinishedProductsInOrder () : void Public  Used to checkout if the products are in order  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatus () : void Public  Prints the number of elements in each queue of the AssemblyLine. This should go into a file, a DB or a log; not to the standard output. However, for the intention of this exercise it will suffice.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| putAfter (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| start () : void Public  Initiates the generation of products and its due threads.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| stop () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

### **Main Class Structure diagram**

*Class diagram in package 'Class Structure'*

Main Class Structure

Version 1.0

Ezequiel H. Martinez created on 6/14/2020. Last modified 6/14/2020



Main Class Structure

### **Main**

*Class in package 'java'*

A simulation of the following use case: "We have to provide a solution for a factory that processes food products in different assembly lines. We need to automate the cooking portion of the assembly line for the products. The factory produces various food products, which are created in assembly lines. There is a common part to many of them, and that is the cooking stages for which some ovens are used. The process to cook the different products (in the â€œcooking stageâ€) in a given point of the assembly line involves getting the products from the line, to put them in the oven for a specific amount of time, in the order they arrive. An intermediate store is used for the products that arrive, if there is no space left in the oven, which has a finite size. If there is no more room in the ovens or the stores when extracting it from the assembly line, the originating assembly line halts. After each product is cooked, we have to extract it and return it to the originating line (this is because multiple lines arrive at this automated stage). We have to develop an application that controls the cooking stage of the factory."

Main

Version 1.0 Phase 1.0 Proposed

hernan ezequiel martinez created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: Main : Class, Public  To: KitchenBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: Main : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLineServer : AssemblyLineServer Private = null  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| getAssemblyLineServer () : AssemblyLineServer Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| main (args : String[] ) : void Public  This whole method, just builds the kitchen and starts to "hear" for requests. This stays looping. If this were an "actual" app, I'll put this hearing requests in a socket. Instead of it, we will work with a Junit that will hit some shared objects and notify it of a new assembly line.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| startAseemblyLineServer () : void Private  Starts a looping thread that heards for request from different assembly lines.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **KitchenBuilder**

*Class in package 'builders'*

Knows how to assemble a kitchen from a .properties file. This is a micture between a Builder and a Factory pattern.

KitchenBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: StoreBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: OvenBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: Main : Class, Public  To: KitchenBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| kitchenBuilder : KitchenBuilder Private = null  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovensToBuild : String Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| PROPERTIES\_SEPARATOR : String Private Const = ","  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| storesToBuild : String Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| buildKitchenStructure () : Kitchen Public  Takes the due properties and builds up a kitchen as it's been requested in the properties files. NOTE: we could have added more properties to configure all the constants in the whole system; we avoided this approach for simplicity.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : KitchenBuilder Public  Singleton kind of implementation as usually a builder/factory pattern could be implemented.  @return  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| KitchenBuilder () : Private  Builds itself from a .properties file  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| readKitchenProperties () : void Private  Loads up how to build the kitchen: how many Ovens and Stores, and their sizes.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **OvenBuilder**

*Class in package 'builders'*

It is in charge of returning an "OvenImpl" kind of object. I do believe these objects could "evolve" during this development, so I try to encapsulate and dettach the way in which we build them.

OvenBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: OvenBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| ovenBuilder : OvenBuilder Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| build (size : Integer ) : OvenImpl Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : OvenBuilder Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| OvenBuilder () : Private  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **StoreBuilder**

*Class in package 'builders'*

We dettach the way we build the stores from the store itself. I do not know if we may want to change the very nature of what a Store 'is' later.

StoreBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: StoreBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| storeBuilder : StoreBuilder Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| build (size : Integer ) : StoreImpl Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : StoreBuilder Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| StoreBuilder () : Private  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **AssemblyLineServer**

*Class in package 'view'*

Receives all the requests from an external source. For this implementation, we will use a unit test. For future use, we will go for the implementation of a REST api (Throught Spring MVC) or hearing in a socket directly. Its major aim is to build the kitchen, start it up. And hear for new assembly lines to appear, dynamically over an external source. Every need assembly line will generate its own products.

AssemblyLineServer

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: AssemblyLineServer : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private Const  This is not how the request will be implemented in a real app!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| endProgram : boolean Private = false  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| kitchen : Kitchen Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| startDateTime : LocalDateTime Private  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private kitchen (Class) Kitchen |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| addAssemblyLine () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| AssemblyLineServer (kitchen : Kitchen ) : Public  Defines the major controllers to operate the simulation. It starts the kitchen and defines the assembly lines.  Properties:  throws = KitchenRequiredException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getActiveSecondsSinceStart () : Long Public  Just returns the amount of seconds since the current object started to hear requests.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getAssemblyLines () : List<AssemblyLine> Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getKitchen () : Kitchen Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isEndProgram () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  We kill the process in cold blood; losing state and data.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printFinishedProductsInOrder () : void Public  This prints in the standard output the products already processed and finished. We use this only in the context of this simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatusAllAssemblyLines () : void Public  We go AssemblyLine by AssemblyLine Asking for the number of Elements in its "waiting" and "finished" lines.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setEndProgram (endProgram : boolean ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| waitForNewAssemblyLines () : void Public  Loops for requests over the creation of new assembly lines (Threads), and starts all the simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **Model Class Structure diagram**

*Class diagram in package 'Class Structure'*

Model Class Structure

Version 1.0

lusip created on 6/14/2020. Last modified 6/14/2020



Model Class Structure

### **AssemblyLine**

*Class in package 'model'*

Represents the implementation of an hypothetical "Food Products Assembly Line" that is about to be simulated.

AssemblyLine

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from AssemblyLine to AssemblyLineStage  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| continueLine : Boolean Private = true  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executor : ExecutorService Private = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| FINISHED\_PRODUCTS\_INITIAL\_CAPACITY : int Private Const = 11  A prime number for good luck. :)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| finishedProducts : PriorityBlockingQueue<Product> Private Const = new PriorityBlockingQueue<>(FINISHED\_PRODUCTS\_INITIAL\_CAPACITY,orderNumberSorter)  OUT queue  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| id : Integer Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_COOK : int Private Const = 15  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_SIZE : int Private Const = 30  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCTIVITY\_DELAY : int Private Const = 3  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_COOK : int Private Const = 5  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_SIZE : int Private Const = 10  DISCLAIMER: These values do not represent REAL cooking times neither food product sizes (just intented for this simulation use).  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCTIVITY\_DELAY : int Private Const = 0  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| orderNumberSorter : Comparator<Product> Package Const = Comparator.comparing(product -> ((Food)product).getOrderNumber())  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| production : AtomicInteger Private = new AtomicInteger(0)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| PRODUCTION\_TIME : long Private Const = 3  I like it every three seconds, a prime number takes the oddities to see some sunshine!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stopProduction : AtomicBoolean Private = new AtomicBoolean(false)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| waitingProducts : ConcurrentLinkedQueue<Product> Private Const = new ConcurrentLinkedQueue<>()  IN queue  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| addProduct (product : Product ) : void Private  Adds the product to the "line" of products that need to be cooked by the Cooker on the multiple ovens.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

|  |
| --- |
| AssemblyLine (id : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| continueProduction () : void Public  Wakes up the production.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| generateRandomProduct () : Product Private  I randomize the values that a new product might have, just before it enters the "input" line. I tries with minimal and maximum values but, ideally, those must be out of some sort of "setup scheme".  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getId () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| haltProduction () : void Public  Stops the production, but the thread continues.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isHalted () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printAllFinishedProductsInOrder () : void Public  Used to checkout if the products are in order  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatus () : void Public  Prints the number of elements in each queue of the AssemblyLine. This should go into a file, a DB or a log; not to the standard output. However, for the intention of this exercise it will suffice.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| putAfter (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| start () : void Public  Initiates the generation of products and its due threads.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| stop () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

### **Food**

*Class in package 'model'*

A generic food. We can extend this class into hamburgers, fries or whatever.

Food

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from Food to Product  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| assemblyLineId : Integer Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| cookTime : Duration Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| orderNumber : Integer Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| size : Integer Private Const  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| cookTime () : Duration Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Food (size : Integer , cookTime : Long ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getAssemblyLineId () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getOrderNumber () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLineId (assemblyLineId : Integer ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setOrderNumber (orderNumber : Integer ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **OvenImpl**

*Class in package 'model'*

It cooks the food and handles its own storage of products

OvenImpl

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from OvenImpl to Oven  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| cookingProducts : CopyOnWriteArrayList<Product> Private Const = new CopyOnWriteArrayList<>()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| on : AtomicBoolean Private = new AtomicBoolean(false)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| size : Integer Private Const  TODO size was Double in the specification, refactor this.  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| OvenImpl (size : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| put (product : Product ) : void Public  Properties:  annotations = @Override  throws = CapacityExceededException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOff () : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn () : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn (duration : Duration ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **StoreImpl**

*Class in package 'model'*

This holds the food, but doesn't cooks it.

StoreImpl

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from StoreImpl to Store  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| size : Integer Private Const  we will not provide a getter & setter for this  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| storedProducts : ConcurrentLinkedQueue<Product> Private Const = new ConcurrentLinkedQueue<>()  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| getSize () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| put (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| StoreImpl (size : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **AssemblyLineStage**

*Interface in package 'model'*

This represents an assembly line stage of the factory. Implementations of this class should be thread-safe

AssemblyLineStage

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from AssemblyLine to AssemblyLineStage  [ Direction is 'Source -> Destination'. ] |

| **OPERATIONS** |
| --- |
| putAfter (product : Product ) : void Public  Put the specified product to the assembly line to continue in the next stage.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Takes the next product available from the assembly line.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **Oven**

*Interface in package 'model'*

This interface represents the OvenImpl that cooks the products in the different assembly lines

Oven

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from OvenImpl to Oven  [ Direction is 'Source -> Destination'. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| put (product : Product ) : void Public  Puts a product in the oven to be cooked. The oven can be functioning at the time the product is put in.  Properties:  throws = CapacityExceededException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  This returns the size of the oven in cm2. As a simplification of the problem, assume that the sizes of the products can be summed, and that value should not exceed the size of the oven. Otherwise an exception is thrown if adding a product.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Take the specified Product out of the oven. The oven can be functioning at the time the product is taken out.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOff () : void Public  Turn off the OvenImpl immediately, even if it was turned on with a duration which will be ignored.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn () : void Public  Turns on the OvenImpl. If the oven was turned on with a duration, the duration is ignored.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn (duration : Duration ) : void Public  Turn on the OvenImpl for the specified duration. If the oven is turned on, it updates the duration.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **Product**

*Interface in package 'model'*

Implementations of this class should take care of overriding the necessary methods of the Object class to allow for the use of Collections in the different implementations of OvenImpl and StoreImpl. This interface is not required to be implemented for this exercise.

Product

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from Food to Product  [ Direction is 'Source -> Destination'. ] |

| **OPERATIONS** |
| --- |
| cookTime () : Duration Public  This is the duration that this product should be cooked for.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  The size that this product physically occupies in cm2  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **Store**

*Interface in package 'model'*

The store where to put the products if the oven is not avialable. This class is thread safe.

Store

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from StoreImpl to Store  [ Direction is 'Source -> Destination'. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| put (product : Product ) : void Public  Put a product in this store, if there is no space left in the store, it will block until enough space frees up. This operation will put the products in FIFO order  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Take the next element that has to be processed respecting FIFO  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Take the specified Product from the StoreImpl  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

### **src**

*Package in package 'Class Structure'*

src

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

#### ***main***

*Package in package 'src'*

main

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

##### **java**

*Package in package 'main'*

java

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

###### ***com***

*Package in package 'java'*

com

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***foodfactory***

*Package in package 'com'*

foodfactory

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

**builders**

*Package in package 'foodfactory'*

builders

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***KitchenBuilder***

*Class in package 'builders'*

Knows how to assemble a kitchen from a .properties file. This is a micture between a Builder and a Factory pattern.

KitchenBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: StoreBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: OvenBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: Main : Class, Public  To: KitchenBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| kitchenBuilder : KitchenBuilder Private = null  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovensToBuild : String Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| PROPERTIES\_SEPARATOR : String Private Const = ","  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| storesToBuild : String Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| buildKitchenStructure () : Kitchen Public  Takes the due properties and builds up a kitchen as it's been requested in the properties files. NOTE: we could have added more properties to configure all the constants in the whole system; we avoided this approach for simplicity.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : KitchenBuilder Public  Singleton kind of implementation as usually a builder/factory pattern could be implemented.  @return  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| KitchenBuilder () : Private  Builds itself from a .properties file  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| readKitchenProperties () : void Private  Loads up how to build the kitchen: how many Ovens and Stores, and their sizes.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***OvenBuilder***

*Class in package 'builders'*

It is in charge of returning an "OvenImpl" kind of object. I do believe these objects could "evolve" during this development, so I try to encapsulate and dettach the way in which we build them.

OvenBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: OvenBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| ovenBuilder : OvenBuilder Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| build (size : Integer ) : OvenImpl Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : OvenBuilder Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| OvenBuilder () : Private  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***StoreBuilder***

*Class in package 'builders'*

We dettach the way we build the stores from the store itself. I do not know if we may want to change the very nature of what a Store 'is' later.

StoreBuilder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: KitchenBuilder : Class, Public  To: StoreBuilder : Class, Public |

| **ATTRIBUTES** |
| --- |
| storeBuilder : StoreBuilder Private  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| build (size : Integer ) : StoreImpl Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getInstance () : StoreBuilder Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| StoreBuilder () : Private  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

**controllers**

*Package in package 'foodfactory'*

controllers

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***Cooker***

*Class in package 'controllers'*

Handles the thread of execution of the Kitchen. What needs to be cooked, in which order. It workd, more or less, like a "business controller" for the backend.

Cooker

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private = null  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovens : List<Oven> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stores : List<Store> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| timers : List<Executor> Private Const = new CopyOnWriteArrayList<>()  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private cooker (Class) Cooker |

| **OPERATIONS** |
| --- |
| addNextFinishedProductToAssemblyLine (idAssemblyLine : Integer , product : Product ) : void Private  Adds one more product to the cache, for its given AssemblyLine  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| cook (product : Product ) : Boolean Public Const  Here lies the main logic of how to put to cook a given product  Properties:  throws = InterruptedException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Cooker (ovens : List<Oven> , stores : List<Store> ) : Public  This constructor allows us to maintain a reference to the Kitchens ovens & stores for the cooker to manage them.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductInOven (product : Product ) : Boolean Private  Tries and places a product in an Oven, if it is successful it starts a timer that simulates the cooking time.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductInStorage (product : Product ) : Boolean Private  Tries to place the product in a Storage, if not, it returns false.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLines (assemblyLines : List<AssemblyLine> ) : void Public  We assign the assemblies lines to have access to the finishedProducts sorted queue.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| startCookingTimer (product : Product ) : void Private  Start a thread with a timer to hold on it; when it finishes, we take the product out of the oven.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOffAllOvens () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

|  |
| --- |
| turnOnAllOvens () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

***Kitchen***

*Class in package 'controllers'*

This is the kitchen, it is composed of many ovens and stores. Assumption: the stores aren't indexed as one per OvenImpl; and the ovens doesn't has stores associated. Number of Ovens: N and Stores: M The kitchen starts the 'cooker' who is a virtual representation of a guy that performs the task of cooking and handling the products to the due assembly lines. There is a requirement: the products should be placed in the line, again in the same order in which they appeared. We solved this with an implementation of a "binary tree" that java has implemented as a "PriorityQueue"; this sort of data structure has the peculiarity of receiving items in any order and then retrieving them ordered. Neat!

Kitchen

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| cooker : Cooker Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| endKitchen : boolean Private = false  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executorCooker : ExecutorService Private Const = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executorDispatcher : ExecutorService Private Const = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| lineContinues : AtomicBoolean Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| ovens : List<Oven> Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stores : List<Store> Private Const  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private cooker (Class) Cooker |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private kitchen (Class) Kitchen |

| **OPERATIONS** |
| --- |
| addOven (oven : Oven ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| addStore (store : Store ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  Brutal! but necessary option...  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Kitchen () : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| placeProductOrHalt (assemblyLine : AssemblyLine , product : Product ) : void Private  It searchs a place for the taken product, if there isn't any...the line halts.  Properties:  throws = InterruptedException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLines (assemblyLines : List<AssemblyLine> ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| start () : void Public  Starts the kitchen's "Cooker". A thread that monitors the assembly lines to take products to cook and retrieve them.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| stop () : void Public  Procedurally STOPS the whole set of processes as is.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| tryGettingProductFromStoresFirst () : Product Private  Before we go for the AssemblyLine, we go into the Stores and checkout for any remaining item to cook!  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

**exceptions**

*Package in package 'foodfactory'*

exceptions

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***CapacityExceededException***

*Class in package 'exceptions'*

CapacityExceededException

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***KitchenRequiredException***

*Class in package 'exceptions'*

KitchenRequiredException

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

**model**

*Package in package 'foodfactory'*

model

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***AssemblyLine***

*Class in package 'model'*

Represents the implementation of an hypothetical "Food Products Assembly Line" that is about to be simulated.

AssemblyLine

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from AssemblyLine to AssemblyLineStage  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| continueLine : Boolean Private = true  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| executor : ExecutorService Private = Executors.newSingleThreadExecutor()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| FINISHED\_PRODUCTS\_INITIAL\_CAPACITY : int Private Const = 11  A prime number for good luck. :)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| finishedProducts : PriorityBlockingQueue<Product> Private Const = new PriorityBlockingQueue<>(FINISHED\_PRODUCTS\_INITIAL\_CAPACITY,orderNumberSorter)  OUT queue  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| id : Integer Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_COOK : int Private Const = 15  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCT\_SIZE : int Private Const = 30  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MAX\_PRODUCTIVITY\_DELAY : int Private Const = 3  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_COOK : int Private Const = 5  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCT\_SIZE : int Private Const = 10  DISCLAIMER: These values do not represent REAL cooking times neither food product sizes (just intented for this simulation use).  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| MIN\_PRODUCTIVITY\_DELAY : int Private Const = 0  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| orderNumberSorter : Comparator<Product> Package Const = Comparator.comparing(product -> ((Food)product).getOrderNumber())  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| production : AtomicInteger Private = new AtomicInteger(0)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| PRODUCTION\_TIME : long Private Const = 3  I like it every three seconds, a prime number takes the oddities to see some sunshine!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| stopProduction : AtomicBoolean Private = new AtomicBoolean(false)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| waitingProducts : ConcurrentLinkedQueue<Product> Private Const = new ConcurrentLinkedQueue<>()  IN queue  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| addProduct (product : Product ) : void Private  Adds the product to the "line" of products that need to be cooked by the Cooker on the multiple ovens.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

|  |
| --- |
| AssemblyLine (id : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| continueProduction () : void Public  Wakes up the production.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| generateRandomProduct () : Product Private  I randomize the values that a new product might have, just before it enters the "input" line. I tries with minimal and maximum values but, ideally, those must be out of some sort of "setup scheme".  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getId () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| haltProduction () : void Public  Stops the production, but the thread continues.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isHalted () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printAllFinishedProductsInOrder () : void Public  Used to checkout if the products are in order  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatus () : void Public  Prints the number of elements in each queue of the AssemblyLine. This should go into a file, a DB or a log; not to the standard output. However, for the intention of this exercise it will suffice.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| putAfter (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| start () : void Public  Initiates the generation of products and its due threads.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| stop () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized True. ] |

***Food***

*Class in package 'model'*

A generic food. We can extend this class into hamburgers, fries or whatever.

Food

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from Food to Product  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| assemblyLineId : Integer Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| cookTime : Duration Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| orderNumber : Integer Private  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| size : Integer Private Const  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| cookTime () : Duration Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| Food (size : Integer , cookTime : Long ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getAssemblyLineId () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getOrderNumber () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setAssemblyLineId (assemblyLineId : Integer ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setOrderNumber (orderNumber : Integer ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***OvenImpl***

*Class in package 'model'*

It cooks the food and handles its own storage of products

OvenImpl

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from OvenImpl to Oven  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| cookingProducts : CopyOnWriteArrayList<Product> Private Const = new CopyOnWriteArrayList<>()  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| on : AtomicBoolean Private = new AtomicBoolean(false)  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| size : Integer Private Const  TODO size was Double in the specification, refactor this.  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| OvenImpl (size : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| put (product : Product ) : void Public  Properties:  annotations = @Override  throws = CapacityExceededException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOff () : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn () : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn (duration : Duration ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***StoreImpl***

*Class in package 'model'*

This holds the food, but doesn't cooks it.

StoreImpl

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **OUTGOING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from StoreImpl to Store  [ Direction is 'Source -> Destination'. ] |

| **ATTRIBUTES** |
| --- |
| size : Integer Private Const  we will not provide a getter & setter for this  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| storedProducts : ConcurrentLinkedQueue<Product> Private Const = new ConcurrentLinkedQueue<>()  [ Is static True. Containment is Not Specified. ] |

| **OPERATIONS** |
| --- |
| getSize () : Integer Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| put (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| StoreImpl (size : Integer ) : Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Properties:  annotations = @Override  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***AssemblyLineStage***

*Interface in package 'model'*

This represents an assembly line stage of the factory. Implementations of this class should be thread-safe

AssemblyLineStage

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from AssemblyLine to AssemblyLineStage  [ Direction is 'Source -> Destination'. ] |

| **OPERATIONS** |
| --- |
| putAfter (product : Product ) : void Public  Put the specified product to the assembly line to continue in the next stage.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Takes the next product available from the assembly line.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***Oven***

*Interface in package 'model'*

This interface represents the OvenImpl that cooks the products in the different assembly lines

Oven

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from OvenImpl to Oven  [ Direction is 'Source -> Destination'. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private ovens (Interface) Oven  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| put (product : Product ) : void Public  Puts a product in the oven to be cooked. The oven can be functioning at the time the product is put in.  Properties:  throws = CapacityExceededException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  This returns the size of the oven in cm2. As a simplification of the problem, assume that the sizes of the products can be summed, and that value should not exceed the size of the oven. Otherwise an exception is thrown if adding a product.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Take the specified Product out of the oven. The oven can be functioning at the time the product is taken out.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOff () : void Public  Turn off the OvenImpl immediately, even if it was turned on with a duration which will be ignored.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn () : void Public  Turns on the OvenImpl. If the oven was turned on with a duration, the duration is ignored.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| turnOn (duration : Duration ) : void Public  Turn on the OvenImpl for the specified duration. If the oven is turned on, it updates the duration.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***Product***

*Interface in package 'model'*

Implementations of this class should take care of overriding the necessary methods of the Object class to allow for the use of Collections in the different implementations of OvenImpl and StoreImpl. This interface is not required to be implemented for this exercise.

Product

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from Food to Product  [ Direction is 'Source -> Destination'. ] |

| **OPERATIONS** |
| --- |
| cookTime () : Duration Public  This is the duration that this product should be cooked for.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| size () : double Public  The size that this product physically occupies in cm2  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

***Store***

*Interface in package 'model'*

The store where to put the products if the oven is not avialable. This class is thread safe.

Store

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **INCOMING STRUCTURAL RELATIONSHIPS** |
| --- |
| Realization from StoreImpl to Store  [ Direction is 'Source -> Destination'. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Kitchen | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Cooker | Target: Private stores (Interface) Store  Cardinality: [0..\*] |

| **OPERATIONS** |
| --- |
| put (product : Product ) : void Public  Put a product in this store, if there is no space left in the store, it will block until enough space frees up. This operation will put the products in FIFO order  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take () : Product Public  Take the next element that has to be processed respecting FIFO  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| take (product : Product ) : void Public  Take the specified Product from the StoreImpl  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

**view**

*Package in package 'foodfactory'*

view

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

***AssemblyLineServer***

*Class in package 'view'*

Receives all the requests from an external source. For this implementation, we will use a unit test. For future use, we will go for the implementation of a REST api (Throught Spring MVC) or hearing in a socket directly. Its major aim is to build the kitchen, start it up. And hear for new assembly lines to appear, dynamically over an external source. Every need assembly line will generate its own products.

AssemblyLineServer

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: AssemblyLineServer : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLines : List<AssemblyLine> Private Const  This is not how the request will be implemented in a real app!  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| endProgram : boolean Private = false  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| kitchen : Kitchen Private Const  [ Is static True. Containment is Not Specified. ] |

|  |
| --- |
| startDateTime : LocalDateTime Private  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private kitchen (Class) Kitchen |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) AssemblyLineServer | Target: Private assemblyLines (Class) AssemblyLine  Cardinality: [0..\*] |

|  |  |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| addAssemblyLine () : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| AssemblyLineServer (kitchen : Kitchen ) : Public  Defines the major controllers to operate the simulation. It starts the kitchen and defines the assembly lines.  Properties:  throws = KitchenRequiredException  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getActiveSecondsSinceStart () : Long Public  Just returns the amount of seconds since the current object started to hear requests.  @return  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getAssemblyLines () : List<AssemblyLine> Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| getKitchen () : Kitchen Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| isEndProgram () : boolean Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| kill () : void Public  We kill the process in cold blood; losing state and data.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printFinishedProductsInOrder () : void Public  This prints in the standard output the products already processed and finished. We use this only in the context of this simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| printStatusAllAssemblyLines () : void Public  We go AssemblyLine by AssemblyLine Asking for the number of Elements in its "waiting" and "finished" lines.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| setEndProgram (endProgram : boolean ) : void Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| waitForNewAssemblyLines () : void Public  Loops for requests over the creation of new assembly lines (Threads), and starts all the simulation.  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

###### ***Main***

*Class in package 'java'*

A simulation of the following use case: "We have to provide a solution for a factory that processes food products in different assembly lines. We need to automate the cooking portion of the assembly line for the products. The factory produces various food products, which are created in assembly lines. There is a common part to many of them, and that is the cooking stages for which some ovens are used. The process to cook the different products (in the â€œcooking stageâ€) in a given point of the assembly line involves getting the products from the line, to put them in the oven for a specific amount of time, in the order they arrive. An intermediate store is used for the products that arrive, if there is no space left in the oven, which has a finite size. If there is no more room in the ovens or the stores when extracting it from the assembly line, the originating assembly line halts. After each product is cooked, we have to extract it and return it to the originating line (this is because multiple lines arrive at this automated stage). We have to develop an application that controls the cooking stage of the factory."

Main

Version 1.0 Phase 1.0 Proposed

hernan ezequiel martinez created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: Main : Class, Public  To: KitchenBuilder : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: Main : Class, Public |

| **ATTRIBUTES** |
| --- |
| assemblyLineServer : AssemblyLineServer Private = null  [ Is static True. Containment is Not Specified. ] |

| **ASSOCIATIONS** | |
| --- | --- |
| Association (direction: Source -> Destination) | |
| Source: Public (Class) Main | Target: Private assemblyLineServer (Class) AssemblyLineServer |

| **OPERATIONS** |
| --- |
| getAssemblyLineServer () : AssemblyLineServer Public  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| main (args : String[] ) : void Public  This whole method, just builds the kitchen and starts to "hear" for requests. This stays looping. If this were an "actual" app, I'll put this hearing requests in a socket. Instead of it, we will work with a Junit that will hit some shared objects and notify it of a new assembly line.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| startAseemblyLineServer () : void Private  Starts a looping thread that heards for request from different assembly lines.  [ Is static True. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

#### ***test***

*Package in package 'src'*

test

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

##### **java**

*Package in package 'test'*

java

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

###### ***MainTest***

*Class in package 'java'*

Executes and test different the creation of new assembly lines and puts the simulation in motion.

MainTest

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

| **CONNECTORS** |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: AssemblyLineServer : Class, Public |

|  |
| --- |
| **Dependency** Source -> Destination  From: MainTest : Class, Public  To: Main : Class, Public |

| **OPERATIONS** |
| --- |
| mainAppStartUpAndStopTest () : void Package  Properties:  annotations = @Test  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

|  |
| --- |
| runAssemblyLinesProductGeneratorTest () : void Package  Properties:  annotations = @Test  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |

## 

## **Interactions**

*Package in package 'Model'*

Interactions

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

### **1. Kitchen Builder diagram**

*Interaction diagram in package 'Interactions'*

1. Kitchen Builder

Version 1.0

lusip created on 6/7/2020. Last modified 6/11/2020



1. Kitchen Builder

### **Note**

*Note in package 'Interactions'*

This stays looping. If this were an "actual" app, I'll put this hearing requests in a socket.

Instead of it, we will work with a Junit that will hit some shared objects and notify it of a new assembly line.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

|  |
| --- |
| **INTERACTION MESSAGES** |
| **1.0 'buildKitchen'** from 'Main Class' sent to 'Kitchen Builder'.  Synchronous Call. Returns Kitchen.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.1 'readKitchenProperties'** from 'Kitchen Builder' sent to 'Kitchen Builder'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.2 'builtKitchenStructure'** from 'Kitchen Builder' sent to 'Kitchen Builder'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.3 'build'** from 'Kitchen Builder' sent to 'Oven's Builder'.  Synchronous Call. Returns void.  [ Return is False. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.4 ''** from 'Oven's Builder' sent to 'Kitchen Builder'.  Synchronous Call. Returns OvenImpl.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.5 'build'** from 'Kitchen Builder' sent to 'Storage's Builder'.  Synchronous Call. Returns void.  [ Return is False. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.6 ''** from 'Storage's Builder' sent to 'Kitchen Builder'.  Synchronous Call. Returns StoreImpl.  [ Return is True. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.7 ''** from 'Kitchen Builder' sent to 'Main Class'.  Synchronous Call. Returns Kitchen.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.8 'startAssemblyLineServer'** from 'Main Class' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.9 'waitForNewAssemblyLines'** from 'Assembly Line Server' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

### **2. Assembly Line Manager diagram**

*Interaction diagram in package 'Interactions'*

2. Assembly Line Manager

Version 1.0

lusip created on 6/7/2020. Last modified 6/11/2020



2. Assembly Line Manager

### **Note**

*Note in package 'Interactions'*

This is here just to illustrate how the rest of the system will interact with the "fake" Assembly Line.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

### **Note**

*Note in package 'Interactions'*

Every AssemblyLine goes in a separated Thread and continues adding at random intervals new products NOTE: We need to define how to "HALT" it.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

|  |
| --- |
| **INTERACTION MESSAGES** |
| **1.0 'createManyAssemblyLines'** from 'Assembly Line Simulator (JUnit)' sent to 'Assembly Line Simulator (JUnit)'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.1 'addAssemblyLine'** from 'Assembly Line Simulator (JUnit)' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is False. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.2 'generateRandomProducts'** from 'Assembly Line Server' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is False. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.3 'addProduct'** from 'Assembly Line Server' sent to 'AssemblyLineProductsQueue'.  Synchronous Call. Returns void.  [ Return is False. Iteration is True. New group is False. ] |

|  |
| --- |
| **1.4 ''** from 'AssemblyLineProductsQueue' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.5 ''** from 'Assembly Line Server' sent to 'Assembly Line Simulator (JUnit)'.  Synchronous Call. Returns AssemblyLine.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.6 'take()'** from 'AssemblyLineStage' sent to 'AssemblyLineProductsQueue'.  Synchronous Call. Returns Product.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.7 ''** from 'AssemblyLineProductsQueue' sent to 'AssemblyLineStage'.  Synchronous Call. Returns Product.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.8 'printStatusAllAssembllyLines'** from 'Assembly Line Simulator (JUnit)' sent to 'Assembly Line Server'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.9 'countCookedItems'** from 'Assembly Line Server' sent to 'AssemblyLineProductsQueue'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.10 ''** from 'AssemblyLineProductsQueue' sent to 'Assembly Line Server'.  Synchronous Call. Returns Integer.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.11 ''** from 'Assembly Line Server' sent to 'Assembly Line Simulator (JUnit)'.  Synchronous Call. Returns void.  [ Return is True. Iteration is False. New group is False. ] |

### **3. Cooker Behavior diagram**

*Interaction diagram in package 'Interactions'*

3. Cooker Behavior

Version 1.0

lusip created on 6/7/2020. Last modified 6/11/2020



3. Cooker Behavior

### **Note**

*Note in package 'Interactions'*

Currently, we cannot use this because the interface doesn't allows it. Can we change that or is an external contral for consumption of a different system?

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

### **Note**

*Note in package 'Interactions'*

This shall start a new Thread with a timer inside of it.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

### **Note**

*Note in package 'Interactions'*

It doesn't has a return type. If this is a "mandatory feature". Then I'll have to resort to return it the same object passed by reference.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

|  |
| --- |
| **INTERACTION MESSAGES** |
| **1.0 'turnOn'** from 'Cooker' sent to 'Oven'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.1 'take()'** from 'Cooker' sent to 'AssemblyLineStage'.  Synchronous Call. Returns Product.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.2 ''** from 'AssemblyLineStage' sent to 'Cooker'.  Synchronous Call. Returns Product.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.3 'startrCooking'** from 'Cooker' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. Life cycle is <none>. ] |

|  |
| --- |
| **1.4 'put'** from 'Cooker' sent to 'Oven'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.5 'take'** from 'Cooker' sent to 'Oven'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.6 ''** from 'Oven' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.7 'take'** from 'Cooker' sent to 'AssemblyLineStage'.  Synchronous Call. Returns Product.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.8 ''** from 'AssemblyLineStage' sent to 'Cooker'.  Synchronous Call. Returns Product.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.9 'startCooking'** from 'Cooker' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.10 'put'** from 'Cooker' sent to 'Oven'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.11 ''** from 'Oven' sent to 'Cooker'.  Synchronous Call. Returns CapacityExceededException.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.12 'storeProduct'** from 'Cooker' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.13 'put'** from 'Cooker' sent to 'Store'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.14 'take'** from 'Cooker' sent to 'AssemblyLineStage'.  Synchronous Call. Returns Product.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.15 ''** from 'AssemblyLineStage' sent to 'Cooker'.  Synchronous Call. Returns Product.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.16 'put'** from 'Cooker' sent to 'Oven'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.17 'CapacityExceededException;'** from 'Oven' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.18 'put'** from 'Cooker' sent to 'Store'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.19 ''** from 'Store' sent to 'Cooker'.  Synchronous Call. Returns IndexOutOfBoundsException.  [ Return is True. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.20 'stopAssemblyLine'** from 'Cooker' sent to 'Cooker'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

|  |
| --- |
| **1.21 '???'** from 'Cooker' sent to 'AssemblyLineStage'.  Synchronous Call. Returns void.  [ Return is False. Iteration is False. New group is False. ] |

### **Assembly Line Simulator (JUnit)**

*Actor in package 'Interactions'*

Assembly Line Simulator (JUnit)

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: createManyAssemblyLines  Sequence from Assembly Line Simulator (JUnit) to Assembly Line Simulator (JUnit) |

|  |
| --- |
| Name: printStatusAllAssembllyLines  Sequence from Assembly Line Simulator (JUnit) to «control» Assembly Line Server |

|  |
| --- |
| Name: addAssemblyLine  Sequence from Assembly Line Simulator (JUnit) to «control» Assembly Line Server |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «control» Assembly Line Server to Assembly Line Simulator (JUnit) |

|  |
| --- |
| Name: createManyAssemblyLines  Sequence from Assembly Line Simulator (JUnit) to Assembly Line Simulator (JUnit) |

|  |
| --- |
| Sequence from «control» Assembly Line Server to Assembly Line Simulator (JUnit) |

### **Cooker**

*Actor in package 'Interactions'*

Cooker

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: take()  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: take  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: startCooking  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: startrCooking  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: take  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: take  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: storeProduct  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: turnOn  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: stopAssemblyLine  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Store |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Store |

|  |
| --- |
| Name: ???  Sequence from Cooker to «boundary» AssemblyLineStage |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: startCooking  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: startrCooking  Sequence from Cooker to Cooker |

|  |
| --- |
| Sequence from «entity» Oven to Cooker |

|  |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

|  |
| --- |
| Name: CapacityExceededException;  Sequence from «entity» Oven to Cooker |

|  |
| --- |
| Name: storeProduct  Sequence from Cooker to Cooker |

|  |
| --- |
| Name: stopAssemblyLine  Sequence from Cooker to Cooker |

|  |
| --- |
| Sequence from «entity» Store to Cooker |

|  |
| --- |
| Sequence from «entity» Oven to Cooker |

|  |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

|  |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

### **Main Class**

*Actor in package 'Interactions'*

Main Class

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: startAssemblyLineServer  Sequence from Main Class to «control» Assembly Line Server |

|  |
| --- |
| Name: buildKitchen  Sequence from Main Class to «control» Kitchen Builder |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «control» Kitchen Builder to Main Class |

### **Assembly Line Server**

*Sequence «control» in package 'Interactions'*

Assembly Line Server

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «control» Assembly Line Server to Assembly Line Simulator (JUnit) |

|  |
| --- |
| Name: countCookedItems  Sequence from «control» Assembly Line Server to «entity» AssemblyLineProductsQueue |

|  |
| --- |
| Name: addProduct  Sequence from «control» Assembly Line Server to «entity» AssemblyLineProductsQueue |

|  |
| --- |
| Sequence from «control» Assembly Line Server to Assembly Line Simulator (JUnit) |

|  |
| --- |
| Name: generateRandomProducts  Sequence from «control» Assembly Line Server to «control» Assembly Line Server |

|  |
| --- |
| Name: waitForNewAssemblyLines  Sequence from «control» Assembly Line Server to «control» Assembly Line Server |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «control» Assembly Line Server |

|  |
| --- |
| Name: startAssemblyLineServer  Sequence from Main Class to «control» Assembly Line Server |

|  |
| --- |
| Name: printStatusAllAssembllyLines  Sequence from Assembly Line Simulator (JUnit) to «control» Assembly Line Server |

|  |
| --- |
| Name: addAssemblyLine  Sequence from Assembly Line Simulator (JUnit) to «control» Assembly Line Server |

|  |
| --- |
| Name: generateRandomProducts  Sequence from «control» Assembly Line Server to «control» Assembly Line Server |

|  |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «control» Assembly Line Server |

|  |
| --- |
| Name: waitForNewAssemblyLines  Sequence from «control» Assembly Line Server to «control» Assembly Line Server |

### **AssemblyLineProductsQueue**

*Sequence «entity» in package 'Interactions'*

AssemblyLineProductsQueue

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «control» Assembly Line Server |

|  |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «boundary» AssemblyLineStage |

|  |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «control» Assembly Line Server |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: countCookedItems  Sequence from «control» Assembly Line Server to «entity» AssemblyLineProductsQueue |

|  |
| --- |
| Name: take()  Sequence from «boundary» AssemblyLineStage to «entity» AssemblyLineProductsQueue |

|  |
| --- |
| Name: addProduct  Sequence from «control» Assembly Line Server to «entity» AssemblyLineProductsQueue |

### **AssemblyLineStage**

*Sequence «boundary» in package 'Interactions'*

AssemblyLineStage

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

|  |
| --- |
| Name: take()  Sequence from «boundary» AssemblyLineStage to «entity» AssemblyLineProductsQueue |

|  |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

|  |
| --- |
| Sequence from «boundary» AssemblyLineStage to Cooker |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: take()  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: take  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Sequence from «entity» AssemblyLineProductsQueue to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: take  Sequence from Cooker to «boundary» AssemblyLineStage |

|  |
| --- |
| Name: ???  Sequence from Cooker to «boundary» AssemblyLineStage |

### **Kitchen Builder**

*Sequence «control» in package 'Interactions'*

Kitchen Builder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: readKitchenProperties  Sequence from «control» Kitchen Builder to «control» Kitchen Builder |

|  |
| --- |
| Name: build  Sequence from «control» Kitchen Builder to «control» Oven's Builder |

|  |
| --- |
| Name: builtKitchenStructure  Sequence from «control» Kitchen Builder to «control» Kitchen Builder |

|  |
| --- |
| Sequence from «control» Kitchen Builder to Main Class |

|  |
| --- |
| Name: build  Sequence from «control» Kitchen Builder to «control» Storage's Builder |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: readKitchenProperties  Sequence from «control» Kitchen Builder to «control» Kitchen Builder |

|  |
| --- |
| Sequence from «control» Oven's Builder to «control» Kitchen Builder |

|  |
| --- |
| Name: builtKitchenStructure  Sequence from «control» Kitchen Builder to «control» Kitchen Builder |

|  |
| --- |
| Sequence from «control» Storage's Builder to «control» Kitchen Builder |

|  |
| --- |
| Name: buildKitchen  Sequence from Main Class to «control» Kitchen Builder |

### **Oven**

*Sequence «entity» in package 'Interactions'*

Oven

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «entity» Oven to Cooker |

|  |
| --- |
| Name: CapacityExceededException;  Sequence from «entity» Oven to Cooker |

|  |
| --- |
| Sequence from «entity» Oven to Cooker |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: take  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: turnOn  Sequence from Cooker to «entity» Oven |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Oven |

### **Oven's Builder**

*Sequence «control» in package 'Interactions'*

Oven's Builder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «control» Oven's Builder to «control» Kitchen Builder |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: build  Sequence from «control» Kitchen Builder to «control» Oven's Builder |

### **Product**

*Sequence «boundary» in package 'Interactions'*

Product

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

### **Storage's Builder**

*Sequence «control» in package 'Interactions'*

Storage's Builder

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «control» Storage's Builder to «control» Kitchen Builder |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: build  Sequence from «control» Kitchen Builder to «control» Storage's Builder |

### **Store**

*Sequence «entity» in package 'Interactions'*

Store

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Sequence from «entity» Store to Cooker |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Name: put  Sequence from Cooker to «entity» Store |

|  |
| --- |
| Name: put  Sequence from Cooker to «entity» Store |

## 

## **State Diagrams**

*Package in package 'Model'*

State Diagrams

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

### **State Diagrams diagram**

*StateMachine diagram in package 'State Diagrams'*

State Diagrams

Version 1.0

lusip created on 6/7/2020. Last modified 6/14/2020



State Diagrams

### **Note**

*Note in package 'State Diagrams'*

Statuses in which we could find a product in the workflow.

Note

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

Extends

### **Final**

*Final State in package 'State Diagrams'*

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Cooked to Final |

### **Synchronization**

*Synchronization in package 'State Diagrams'*

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Synchronization to In a Store |

|  |
| --- |
| Transition from Synchronization to Cooking |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from In Assembly Line to Synchronization |

### **Cooked**

*State in package 'State Diagrams'*

Cooked

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Cooked to Final |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Cooking to Cooked |

### **Cooking**

*State in package 'State Diagrams'*

Cooking

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Cooking to Cooked |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Synchronization to Cooking |

|  |
| --- |
| Transition from In a Store to Cooking |

### **In a Store**

*State in package 'State Diagrams'*

In a Store

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from In a Store to Cooking |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Synchronization to In a Store |

### **In an Assembly Line**

*State in package 'State Diagrams'*

In an Assembly Line

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

### **In Assembly Line**

*State in package 'State Diagrams'*

In Assembly Line

Version 1.0 Phase 1.0 Proposed

lusip created on 6/7/2020. Last modified 6/7/2020

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from In Assembly Line to Synchronization |

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Initial to In Assembly Line |

### **ExitPoint**

*ExitPoint in package 'State Diagrams'*

### **Final**

*Final State in package 'State Diagrams'*

| **INCOMING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Cooked to Final |

### **Initial**

*Initial State in package 'State Diagrams'*

| **OUTGOING BEHAVIORAL RELATIONSHIPS** |
| --- |
| Transition from Initial to In Assembly Line |

## **Food\_Factory\_Introduction\_Document**

*Artifact «Document» in package 'Model'*

Food\_Factory\_Introduction\_Document

Version 1.0 Phase 1.0 Proposed

lusip created on 6/14/2020. Last modified 6/14/2020

## **Source Code Review**

|  |  |  |  |
| --- | --- | --- | --- |
| **Programmer:** |  | **Reviewed by:** |  |
| **Language:** |  | **Date:** |  |
| **Version:** |  | **Phase:** |  |

***Naming Conventions***

|  |  |  |  |
| --- | --- | --- | --- |
| **Areas** | **Modules** | **Accepted** | **Comments** |
| **Classes:** | All | ü / û |  |
| **Variables:** |  |  |  |
| **Files:** |  |  |  |
| **Namespaces:** |  |  |  |
|  |  |  |  |

***Code Construction***

|  |  |  |  |
| --- | --- | --- | --- |
| **Areas** | **Modules** | **Accepted** | **Comments** |
| **Classes:** |  | ü / û |  |
| **Functions:** |  |  |  |
| **Macros:** |  |  |  |
|  |  |  |  |

***Commenting Style***

|  |  |  |  |
| --- | --- | --- | --- |
| **Areas** | **Modules** | **Accepted** | **Comments** |
| **Header Comments:** |  | ü / û |  |
| **Line Comments:** |  |  |  |
| **Details:** |  |  |  |
|  |  |  |  |

***Formatting Style***

|  |  |  |  |
| --- | --- | --- | --- |
| **Areas** | **Modules** | **Accepted** | **Comments** |
| **Indentation:** |  | ü / û |  |
| **Spacing:** |  |  |  |
| **Bracketing:** |  |  |  |
|  |  |  |  |

|  |
| --- |
| **General Comments:**  *<Provide any comments here on the adherence to the standards as laid out above. There must be a clear differentiation between the standards as opposed to conventions seen as best practices. In most cases the standards must be adhered to. Where there is deviation on this it should be documented here. Best practice conventions are open to variation so long as these meet the required structure. >* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signed by:** |  |  | / / |  |
|  | Name |  | Date |  |