Requerimiento Funcionales

- El cliente debe poder obtener una lista de códigos ISBN de los ejemplares que desea obtener.
- Al ingresar los códigos ISBN a la máquina y ésta le debe indicar al cliente los bloques o estanterías en donde debe buscar el volumen de su interés.
- Se le debe asignar un canasto al cliente para ir ubicando los libros que va encontrando, uno encima del otro.
- Teniendo en cuenta el tiempo en el que ha entrado a la tienda, más lo que se ha tomado recogiendo los libros de cada estantería, se debe establecer el orden con el que se ingresa a la fila de cajas.
- El último libro añadido debe ser el primero facturado y posteriormente empacado.

Diseño de Casos de Prueba

Clase	Método	Escenario	Valores de Entrada	Resultado
Stack	getSize()	Se crea un objeto tipo Stack, este tiene dos libros	Libro1 Libro2	El tamaño de la Stack es 2 exitosamente.
Stack	isEmpty()	Se crea un objeto tipo Stack, este tiene dos libros	Libro1 Libro2	La Stack no está vacía exitosamente.
Stack	isEmpty()	Se crea un objeto tipo Stack.	Ninguno	La Stack está vacía exitosamente.
Stack	peek()	Se crea un objeto tipo Stack.	Ninguno	El método lanza una excepción exitosamente
Stack	peek()	Se crea un objeto tipo Stack, este tiene dos libros	Libro1 Libro2	El método retorna el primer elemento exitosamente

Stack	pop()	Se crea un objeto tipo Stack.	Ninguno	El método lanza una excepción exitosamente
Stack	pop()	Se crea un objeto tipo Stack, este tiene dos libros	Libro1 Libro2	El método retorna y elimina el primer elemento exitosamente
Stack	push()	Se crea un objeto tipo Stack, este tiene dos libros	Libro1 Libro2	Los libros son guardados exitosamente.
Queue	isEmpty()	Se crea un objeto tipo Queue, este tiene dos clientes	Cliente1 Cliente2	La Queue no está vacía exitosamente.
Queue	isEmpty()	Se crea un objeto tipo Queue.	Ninguno	La Queue está vacía exitosamente.
Queue	offer()	Se crea un objeto tipo Queue, este tiene dos clientes	Cliente1 Cliente2	Los clientes se han añadido exitosamente
Queue	peek()	Se crea un objeto tipo Queue.	Ninguno	El método lanza una excepción exitosamente
Queue	peek()	Se crea un objeto tipo Queue, este tiene dos clientes	Cliente1 Cliente2	El método retorna el primer elemento exitosamente

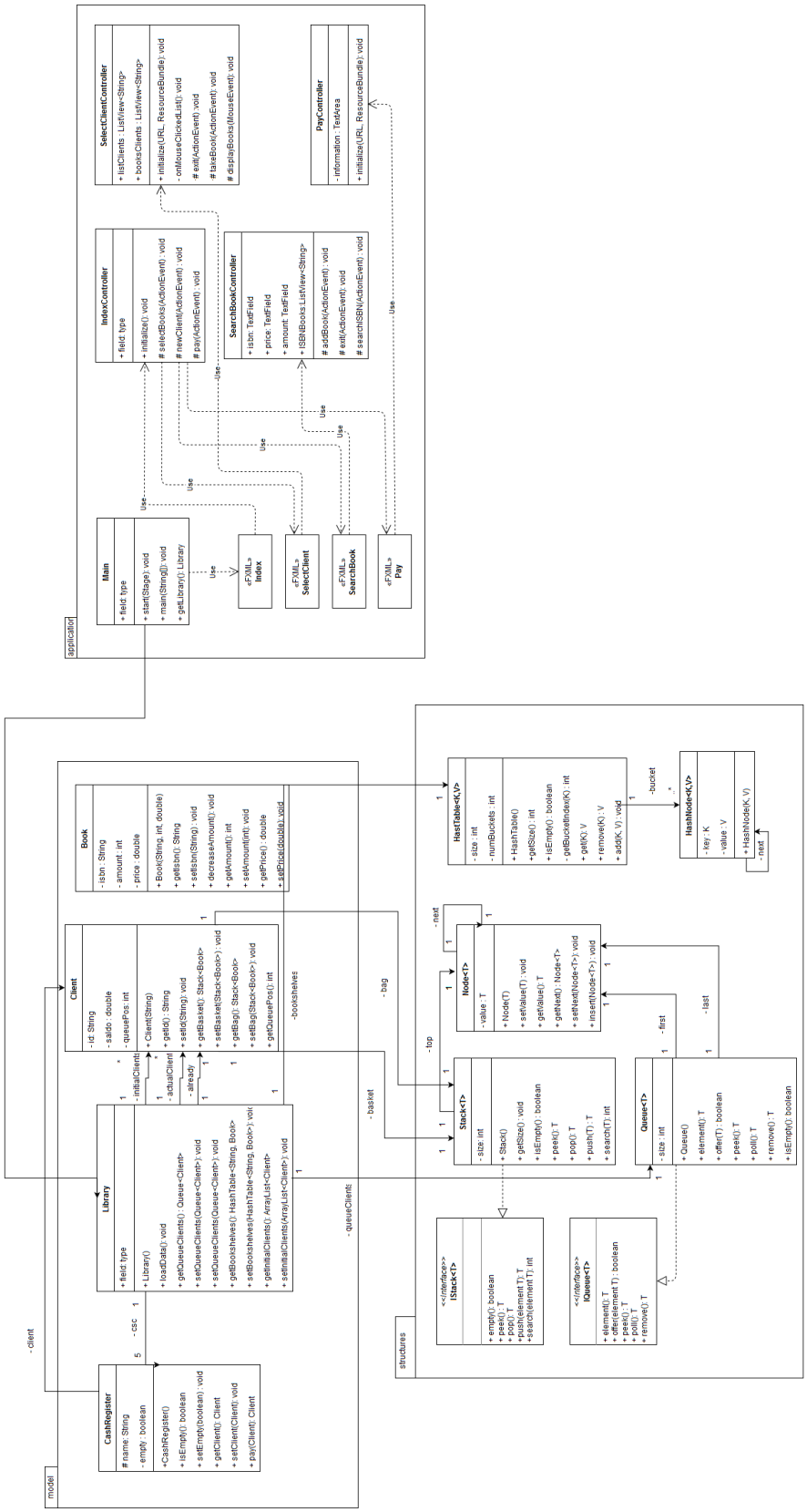
Queue	poll()	Se crea un objeto tipo Queue.	Ninguno	El método lanza una excepción exitosamente
Queue	poll()	Se crea un objeto tipo Queue, este tiene dos clientes	Cliente1 Cliente2	El método retorna y elimina el primer elemento exitosamente
HashTable	isEmpty()	Se crea un objeto tipo HashTable, este tiene dos libros con sus respectivas llaves	Id1 = “12” Id2 = “122” Cliente1 Cliente2	La HashTable no está vacía exitosamente.
HashTable	isEmpty()	Se crea un objeto tipo HashTable.	Ninguno	La HashTable está vacía exitosamente.
HashTable	add()	Se crea un objeto tipo HashTable, este tiene dos libros con sus respectivas llaves	Id1 = “12” Id2 = “122” Cliente1 Cliente2	Los libros se han guardado exitosamente.
HashTable	get()	Se crea un objeto tipo HashTable, este tiene dos libros con sus respectivas llaves	Id1 = “12”	El método retorna el Libro correspondiente al id = “12”

HashTable	getSize()	Se crea un objeto tipo HashTable, este tiene dos libros con sus respectivas llaves	Id1 = “12” Id2 = “122” Cliente1 Cliente2	El tamaño de la HashTable es 2 exitosamente.
HashTable	remove()	Se crea un objeto tipo HashTable, este tiene dos libros con sus respectivas llaves	Id1 = “12”	El libro con id = “12” es removido exitosamente

TAD Queue
FIFO(First In FIrst Out)
Sólo se puede modificar la cabeza de la Queue
Operaciones **-**peek - Queue -> Queue -poll -Queue -> Queue -offer -Queue -Element -> Queue -isEmpty -Queue -> boolean

TAD Stack
FIFO(First In Last Out)
Sólo se puede modificar la cabeza de la Stack
Operaciones **-**peek - Stack -> Stack -pop -Stack -> Stack -push -Stack -Element -> Stack -isEmpty -Stack-> boolean

Algoritmos y Estructuras
Laboratorio II



Algoritmos y Estructuras
Laboratorio II