UNIVERSTIY OF WESTMINSTER
INFORMATICS INSTITUTE OF TECHNOLOGY

MODULE: 5COSC019C.1 Object Oriented Programming
Module Leader: Guhanathan Poravi

Submission: CW (Real Time Event Ticketing Systems)

Documentation (Test Cases, Class Diagrams, Sequence Diagrams)

Name: Themiya Jayaratne

UOW: w2084732
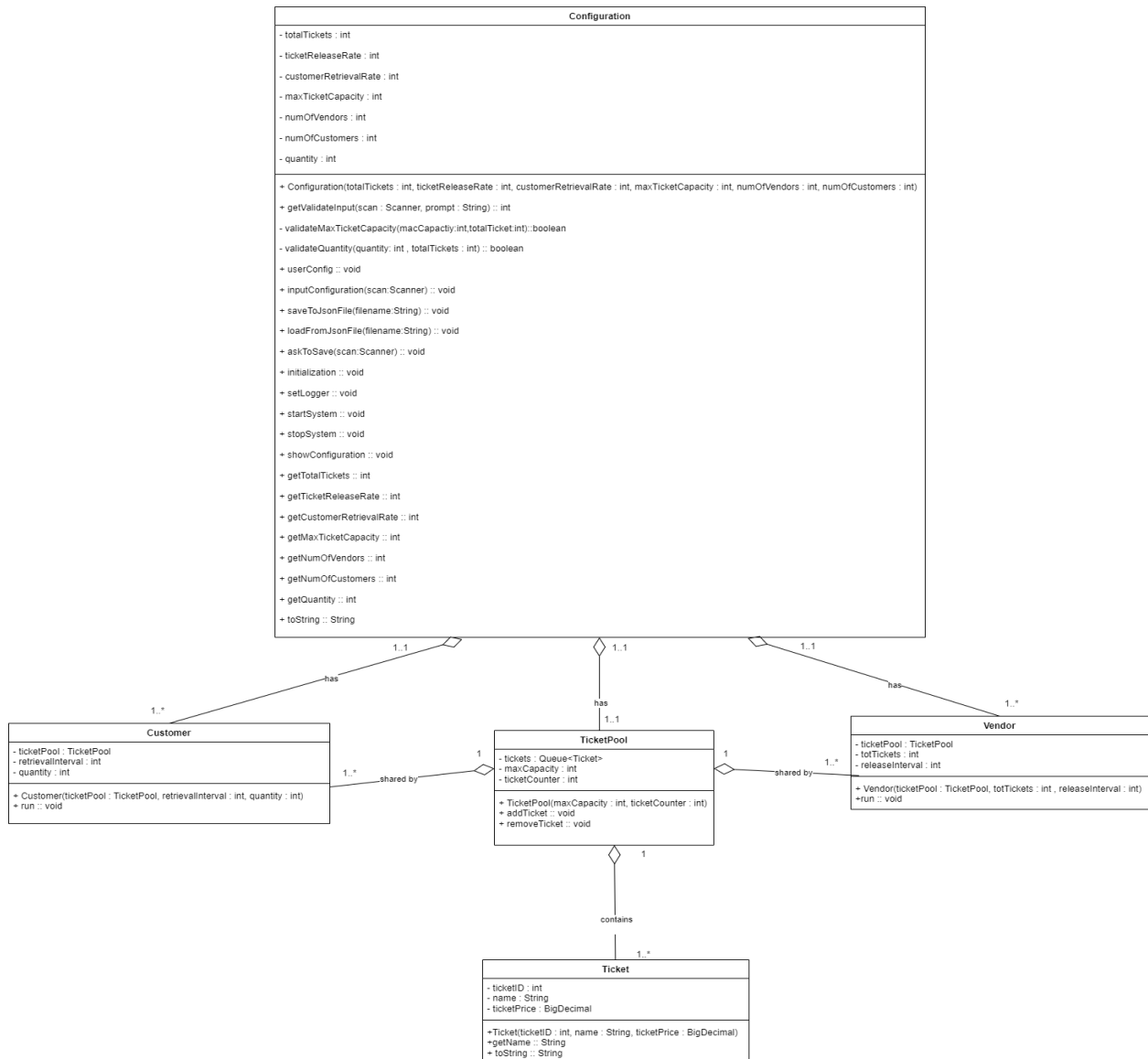
IIT ID: 20232689

CS group 02

Contents

1. Test Cases (CLI)

Test ID		Expected Result	Actual Output	Status
TC1	Start system with no configuration	Message displayed in console "System is not configured"	Message displayed in console notifying user the system is not configured	PASS
TC2	Enter values less than 0 for configuration fields	Message displayed in console "Only positive values"	Message displayed in console notifying user that only positive values are accepted	PASS
TC3	Enter invalid data type for configuration	Prompt user to enter the value again	Message displayed in console re-prompting user to enter the value again	PASS
TC4	Enter max ticket capacity greater than total tickets	Error message displayed notifying users to enter an amount less than total tickets	Message displayed to user notifying max ticket capacity should be less than total tickets	PASS
TC5	Enter customer ticket buying quantity greater than total tickets	Error message displayed notifying user to enter an amount less than total tickets	Message displayed to user notifying quantity should be less than total tickets	PASS
TC6	Start non-running system	Warn user that the system is not running	Message displayed to user notifying the system is not running	PASS
TC7	Enter invalid choice out of menu scope	Notify the option is invalid	User notified the option is invalid	PASS
TC8	Save configuration to a Json file	Ask user to save configuration with file name and save configuration	File created	PASS
TC9	Load configuration from Json file	Configuration fields loaded from Json file	Configuration loaded	PASS

TC10	Save System logs to a text file	Save logging information to a file	File created and logging information saved	PASS
TC11	Start System	Vendor and customer threads start	Vendor and Customer initialized and starting	PASS
TC12	Stop System	Vendor and customer threads interrupted and joined	Vendor and customer threads stop	PASS
TC13	addTicket() method for vendor	Run method in Vendor should add tickets to the ticket pool	Vendor threads add tickets to the ticket pool	PASS
TC14	removeTicket() method for customer	Run method in Customer should remove tickets from the ticket pool	Customer threads retrieving tickets from the ticket pool	PASS
TC15	Proper logging display of Vendor Customer ticket interactions	Vendor producing ticket message and customer buying message displayed in console	Number of vendors and customers interacting with tickets properly displayed	PASS
TC16	Vendors producing tickets	Vendor will produce tickets up to the total number of tickets	Vendor produces tickets up to the total number of tickets while waiting until the ticket pool is empty	PASS
TC17	Customer buying tickets	Each customer will buy tickets according to the given quantity	Each customer buys tickets up to allocated quantity	PASS
TC18	System termination	User enter a value and the system exits	Press 5 and the system terminates	PASS

2. Class Diagrams (Main classes)

Configuration

- totalTickets : int
- ticketReleaseRate : int
- customerRetrievalRate : int
- maxTicketCapacity : int
- numOfVendors : int
- numOfCustomers : int
- quantity : int

- + Configuration(totalTickets : int, ticketReleaseRate : int, customerRetrievalRate : int, maxTicketCapacity : int, numOfVendors : int, numOfCustomers : int)
- + getValidateInput(scan : Scanner, prompt : String) :: int
- - validateMaxTicketCapacity(macCapactiy:int,totalTicket:int)::boolean
- - validateQuantity(quantity: int , totalTickets : int) :: boolean
- + userConfig :: void
- + inputConfiguration(scan:Scanner) :: void
- + saveToJsonFile(filename:String) :: void
- + loadFromJsonFile(filename:String) :: void
- + askToSave(scan:Scanner) :: void
- + initialization :: void
- + setLogger :: void
- + startSystem :: void
- + stopSystem :: void
- + showConfiguration :: void
- + getTotalTickets :: int
- + getTicketReleaseRate :: int
- + getCustomerRetrievalRate :: int
- + getMaxTicketCapacity :: int
- + getNumOfVendors :: int
- + getNumOfCustomers :: int
- + getQuantity :: int
- + toString :: String

Customer

- ticketPool : TicketPool
- retrievalInterval : int
- quantity : int

- + Customer(ticketPool : TicketPool, retrievalInterval : int, quantity : int)
- + run :: void

TicketPool

- tickets : Queue<Ticket>
- maxCapacity : int
- ticketCounter : int

- + TicketPool(maxCapacity : int, ticketCounter : int)
- + addTicket :: void
- + removeTicket :: void

Vendor

- ticketPool : TicketPool
- totTickets : int
- releaseInterval : int

- + Vendor(ticketPool : TicketPool, totTickets : int , releaseInterval : int)
- + run :: void

Ticket

- ticketID : int
- name : String
- ticketPrice : BigDecimal

- +Ticket(ticketID : int, name : String, ticketPrice : BigDecimal)
- +getName :: String
- + toString :: String

has (1..1) · *shared by* (1..*) · *contains* (1..*)

3. Sequence Diagram (Main Classes)