Kyle O'Dell

824811891

Project file consists of five packages: exceptions, data, driver, interfaces, stackqueue.

Below is are the packages and it's respective classes (their in depth details are listed on the following pages):

Exceptions (contains custom exceptions)

-EmptyStackQException.java

data (only holds the given DataClass)

-DataClass.java

driver

-Driver.java (Where StackQ is tested)

interfaces (holds all interfaces)

-QueueSpecs.java (Specifications for a Queue)

-StackSpecs.java (Specifications for a Stack)

stackqueue (holds all objects needed to create StackQ)

-LLStack.java (Stack implemented on a linkedlist)

-StackQ.java (Queue made out of two LLStack's)

-Node.java (Objected needed to create a linkedlist)

EmptyStackQException.java

-Class extends the Exception class and is designed to throw a custom exception for when a StackQ is empty.

Public constructors:

-EmptyStackQException(): initializes exception by sending “StackQ is empty!” through super constructor

---

LLStack.java

A stack implemented on a linked list

-implements StackSpecs interface

private member variables:

-Node<E> top: references the top of the LLStack.

-int stackSize: keeps track of how many items are in the LLStack.

Constructors:

-LLStack(): default constructor, initializes an empty LLStack. The top node is set to null, and the stackSize is set to zero.

-LLStack(E obj): initializes a new LLStack with the parameter object of the top of the stack. Sets the top node to a node with the parameter object within, and sets the stackSize to one.

Public member methods:

-void setTop(Node<E> top): sets this.top to the parameter node called “top”.

-Node<E> getTop(): returns this.top variable.

-void setStackSize(int stackSize): this the this.stackSize variable to the parameter stackSize.

-int getStackSize(): returns the stackSize variable.

-boolean isEmpty(): initializes boolean variable called “isEmpty” to false, if the stackSize variable equals zero, then the isEmpty variable is set to true and then the isEmpty variable is returned.

-void emptyStack(): keeps using this.pop() until the stack is empty.

-void push(E obj): initializes a new Node<E> object with the parameter object. If the stackSize variable is zero, sets the top node to the new node. Else, sets the new nodes next node to the top node of the stack, then sets the top node to the new node. Then increments the stackSize variable by one.

-E pop(): Sets a generic variable called “removed” to null, then declares a Node<E> called “temp”. if this stack is not empty, then if the stackSize variable equals one, then the removed variable gets set to the data of the top node and the top node is set to null. Else, the removed variable gets set to the top node data, then the temp node is set to the tops next node, then the top node is set to null, then the top node is set to the temp node, then the temp node is set to null. Then the stackSize variable is decremented by one. If this stack is empty, null is returned.

-E peek(): Sets a generic variable called “peeked” to null. If the top node is not null, then the peeked variable is set to the top nodes data and is returned. If the top node is null, then null is returned.

-String toString(): initializes a LLStack<E> called “temp”, then initializes a string called “contents” with “[(Bottom) “. Then, while this stack is not empty, the temp stack pushes all of this stack’s popped objects. Then, while the temp stack is not empty, the contents string is concatenated with each peeked object from the temp stack, this stack pushes the temp stack’s popped object, and if the temp stack is not empty, it concatenates “, “ onto the contents string. Finally, the contents string is concatenated with “ (Top)]”, and the string is returned.

Node.java

Generic object holder designed for building a linkedlist

Private member variables:

-E data: the object the node holds

-Node<E> nextNode: reference to the next node

Public constructors:

-Node(E data): initializes node with parameter object, points nextNode to null. this.data is set to the parameter data, and this.nextNode is set to null.

-Node(E data, Node<E> node): initializes node with parameter object, points nextNode to parameter node. this.data is set to the parameter data, and this.nextNode is set to the parameter node.

Public member methods:

-E getData: returns the this.data variable.

-void setData(E data): sets this.data to parameter data.

-Node<E> getNextNode(): returns this.nextNode variable.

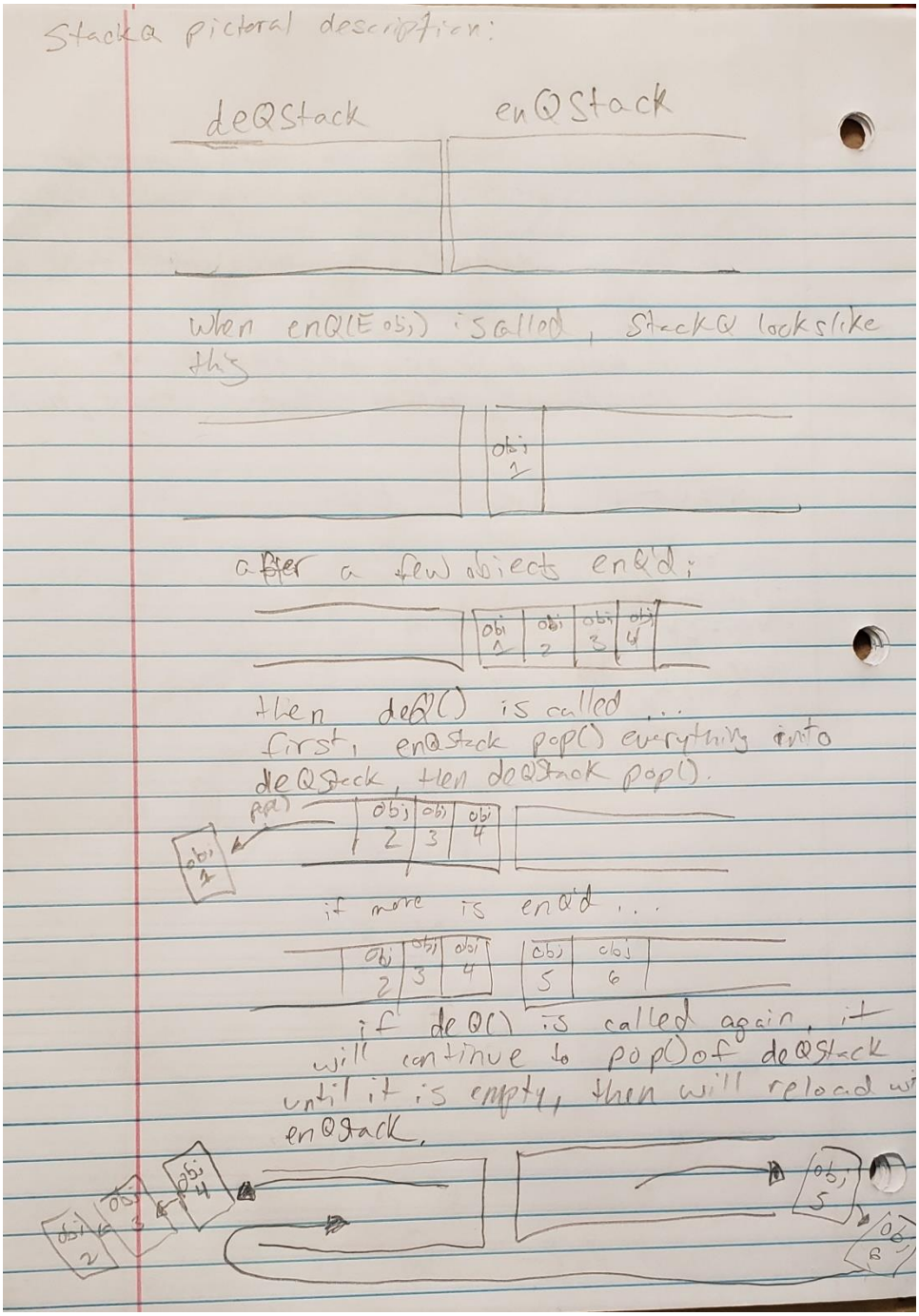
-void setNextNode(Node<E> nextNode): sets this.nextNode to the parameter node.

(Next page contains info and pictorial description of StackQ.java)

StackQ.java

A queue made out of two stacks

-implements QueueSpecs interface



Private member variables:

-LLStack<E> enQStack: stack that holds the objects that are enqueued

-LLStack<E> deQStack: stack that holds the objects that are being dequeued

Public constructors:

-StackQ(): default constructor, initializes new private stacks. Uses its own constructor to initialize enQStack to a new LLStack and deQStack to a new LLStack.

-StackQ(LLStack<E> es, LLStack<E> ds): allows user to set private stacks to already existing stacks. enQStack gets set to the es LLStack and deQStack gets set to the ds LLStack.

Public member methods:

-boolean isEmpty(): initializes a boolean variable called “isEmpty” to false. If enQStack is empty and deQStack is empty, the isEmpty is set to true. Then isEmpty is returned.

-void emptyQueue(): uses enQStack.emptyStack() then uses deQStack.emptyStack() in order to empty the queue.

-void enQ(E obj): uses enQStack.push(obj) to push the parameter object onto the enQStack.

-E deQ(): Initializes a generic object called “deQd” to null. If deQStack is empty, then while the enQStack is not empty, pops all of the enQStack objects onto the deQStack. Then deQStack pops an item into the variable deQd and deQd is returned.

-E peek(): Initializes a generic object called “peeked” to null. If deQStack is empty, then while the enQStack is not empty, pops all of the enQStack objects onto the deQStack. Peeked is set to the peeked object off of deQStack. Then peeked is returned.

-LLStack<E> getdeQStack(): returns reference to deQStack.

-void setdeQStack(LLStack<E> deQStack): sets this.deQStack to parameter stack.

-LLStack<E> getenQStack(): returns reference to enQStack.

-void setenQStack(LLStack<E> enQStack): sets this.enQStack to parameter stack.

-int getQueueSize(): returns the size of the queue by adding up the size of enQStack and deQStack.

-String toString(): initializes a StackQ<E> called “temp”. initializes a String called “contents” with “[(Front) “. While this StackQ is not empty, temp enQ’s all of this StackQ’s deQ’d objects. Then while temp StackQ is not empty, concatenate contents with temp StackQ’s peeked object, then

this StackQ enQ's the temp StackQ's deQ'd object, then if the temp StackQ is not empty, concatenate contents with ", ". Finally, concatenate contents with " (Rear)]", then return contents.

StackSpecs.java

Generic interface contains the following declared methods that are desired to create a Stack:

-public Boolean isEmpty();

-public void emptyStack();

-public void push(E obj);

-public E pop();

-public E peek();

QueueSpecs.java

Generic interface contains the following declared methods that are desired to create a Queue:

-public boolean isEmpty();

-public void emptyQueue();

-public void enQ(E obj);

-public E deQ();

-public E peek();

DataClass.java

Private member variables:

-String dataName: stores name of data

-int dataID: stores ID of data

Public constructors:

-DataClass(String dataName, int dataID): calls super constructor, then sets this.dataName to parameter dataName, then sets this.dataID to parameter dataID.

Public member methods:

-String getDataName(): returns this.dataName.

-void setDataName(String dataName): sets this.dataName to parameter dataName.

-int getDataID(): returns this.dataID.

-void setDataID(int dataID): sets this.dataID to parameter dataID.

-String toString(): returns a string in the format of "[" concatenated with this.data name, ", ", this.dataID and "]".

Driver.java

Where the main method is located.

Public static void main(String[] args): Initializes a Scanner called "in" (used to pull input from keyboard). Initializes two int's called "input" and "datagen", both are set to zero. Initializes a boolean called "menuActive" to true. Initializes a StackQ<DataClass> called "que". A do-while loop is performed while variable menuActive is true. A group of System.out.println(String message) statements are called that display a menu. Scanner in will then wait for user input, where a try-catch block will wait for an input that is not an int. if input is not an int, an exception is thrown and caught, displaying a message to the console and repeats the menu. If the user inputs 1, then a generated DataClass object is enqueued onto StackQ que. Else, if user inputs 2, the StackQ que dequeues and displays object to the console. Else, if user inputs 3, StackQ que sends a peeked object to the console. Else, if user inputs 4, StackQ que calls its .toString(). Else, if user inputs 5, StackQ que calls its deQStack and enQStack .toString(). Else, if user inputs 6, StackQ que calls its .getQueueSize(). Else, if user inputs 7, the menuActive variable is set to false, and the menu quits. Goodbye message is displayed to console.