Alexander Chatron-Michaud

260611509

QUESTION 1:

Student.java:

/\*\*

\* Class for a student with an age, name, and student ID.

\* @author Alexander Chatron-Michaud

\*/

public class Student {

private int age;

private String name;

private int id;

public Student(int init\_age, String init\_name, int init\_id) {

if ((init\_id > 99999) || (init\_id < 10000) || (init\_age < 5)) {

System.out.println("Error: Student ID is not 5 digits or student is under 5 years old");

throw new IllegalStateException();

}

age = init\_age;

name = init\_name;

id = init\_id;

}

/\*\*

\*Returns the ID of the student

\*@return Integer form ID of the student

\*/

public int get\_id() {

return this.id;

}

/\*\*

\*Returns the age of the student

\*@return Integer form age of the student

\*/

public int get\_age() {

return this.age;

}

Queue.java:

import java.util.\*;

/\*\*

\* Class to manage a lineup of students entering a gym party

\* @author Alexander Chatron-Michaud

\*/

public class Queue {

private LinkedList<Student> line;

public Queue() {

line = new LinkedList<Student>();

}

/\*\*

\*Adds a student to the line if they are a valid student

\*@param a the student to be added

\*/

public void enqueue(Student a) {

if (isValid(a)) {

line.addLast(a);

}

}

/\*\*

\* If there is a student in line, remove the first student in line

\* so they can enter the party.

\* @return the student to enter the party, or null if line is empty

\*/

public Student dequeue() {

if (line.size() == 0) {

System.out.println("Error: No students in queue to dequeue");

throw new IllegalStateException();

}

return line.removeFirst();

}

/\*\*

\*Checks if a student is valid to enter the party.

\*A student is valid if their 5 digit

\*ID begins with 22 and they are at least 15 years old.

\*@param a the student in question

\*@return True if the student is valid to enter party, False is the student is not valid to enter party

\*/

public boolean isValid(Student a) {

if ((a.get\_age() >= 15) && (a.get\_id() >= 22000) && (a.get\_id() < 23000)){

return true;

}

else {

return false;

}

}

}

QUESTION 2:



USE CASE TEMPLATE:

Name: ABC boards a passenger onto a plane

Creator: Alexander Chatron-Michaud

Description:

1. Login to Management
2. Access passenger management
3. Check if the passenger has a valid boarding pass
4. Set validity of boarding pass
5. Board passenger onto plane

Special cases:

* Potential failure moving out of step 1 if login failed
* Potential failure if boarding pass is invalid
* When attempting to board a passenger, potential failure cases
  + Not enough employees to run airline legally
  + Plane does not exist
  + Route does not exist
  + Wrong plane or route with respect to boarding pass or plane
  + Plane is not located in airport for departure

QUESTION 3:



QUESTION 4:

