CSC 151 Assignment #2

1. Honor Code

A. For individual assignments: Jane Doe and John Doe will be replaced by your full name(s)

I affirm that I have carried out my academic endeavors with full academic honesty.

[Signed, Jane Doe]

For group assignments (when allowed):

James Heffernan

Manav Bilakhia

Saeed AlSuwaidi

Eric Zhao

B. Resources/References

2. Java files and outputs

A. Java files

```
/*
 * Honor Code
 */
package assignment;

public class TShirts {
    //0:S 1:M 2:L 3:XL 4:Big Size You may use a String array
    private int size;
    static int MAX_SIZE = 4;
    private String color;
    static int MIN_SIZE = 0;
    //Constant values for MAX_SIZE and MIN_SIZE

    //Instance variables

    /**
     * @param size
     * @param color
     */
    //Constructor
    public TShirts(int size, String color) {
        this.setColor(color);
        this.setSize(size);
    }
    //Set/get methods

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        if (size > MAX_SIZE || size < MIN_SIZE) {
            this.size = MIN_SIZE;
```

```
    } else {
        this.size = size;
    }
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    if (color == "Red" || color == "Blue" || color == "Green" || color == "red" || color == "blue" || color == "green") {
        this.color = color;
    } else {
        this.color = "";
    }
}

public void setAll(int size, String color) {
    setColor(color);
    setSize(size);
}

//toString method
@Override
public String toString() {
    switch (this.size) {
        case 0:
            return "size=S " + "color=" + color + "\n";
        case 1:
            return "size=M " + "color=" + color + "\n";
        case 2:
            return "size=L " + "color=" + color + "\n";
        case 3:
            return "size=XL " + "color=" + color + "\n";
        case 4:
            return "size=Big " + "color=" + color + "\n";
        default:
            return "you shouuldnt be here!";
    }
}

//main method. Please change this completely
public static void main(String[] args) {
    String colors[] = {"Red", "Green", "Blue" };
    TShirts tsh[] = new TShirts[4];
    for (int i = 0; i < tsh.length; i++)
        tsh[i] = new TShirts(i, colors[i % 3]);
    for (int i = 0; i < tsh.length; i++)
        System.out.print(tsh[i]);
```

```
    }
}
```

```
package assignment;

import org.junit.Assert;

import static org.junit.jupiter.api.Assertions.*;

class TShirtsTest {

    @org.junit.jupiter.api.Test
    void getSize() {
        TShirts tsh = new TShirts(2,"green");
        assertEquals(2,tsh.getSize());

    }

    @org.junit.jupiter.api.Test
    void getColor() {
        TShirts tsh = new TShirts(2,"green");
        assertEquals("green",tsh.getColor());
    }

    @org.junit.jupiter.api.Test
    void setAll() {
        TShirts tsh = new TShirts(2,"green");
        tsh.setAll(3,"red");
        assertEquals("red",tsh.getColor());
        assertEquals(3,tsh.getSize());

    }
}
```

```
package assignment;

import java.util.Arrays;
import java.util.Stack;
import java.util.StringJoiner;

public class TShirtStack {
//0:S 1:M 2:L 3:XL 4:Big Size You may use a String array

    //Constant values for STOCK_LIMIT
    private static int STOCK_LIMIT = 3;
    private Stack<TShirts> red;
    private Stack<TShirts> blue;
    private Stack<TShirts> green;
    private Integer[] sizeInStock; // 0 = small. 1 = mediunm, 2 = large, 3 = XL, 4 = big
    private Integer[] soldOut;

    //Instance variables red, green, blue Stacks
    //and int arrays sizesInStock and soldOut
```

```
//Parameterless Constructor initialize arrays and stacks
public TShirtStack() {
    red = new Stack<>();
    blue = new Stack<>();
    green = new Stack<>();
    sizeInStock = new Integer[5];
    soldOut = new Integer[5];
    for (int i = 0; i < sizeInStock.length; i++) {
        sizeInStock[i] = 0;
        soldOut[i] = 0;
    }
}

//addTShirt method
public void addTShirt(TShirts TShirt) {
    String color = TShirt.getColor();
    if(color.equalsIgnoreCase("red"))
    {
        red.push(TShirt);
        sizeInStock[TShirt.getSize()] += 1;
    }
    else if (color.equalsIgnoreCase("blue")){
        blue.push(TShirt);
        sizeInStock[TShirt.getSize()] += 1;
    }
    else if(color.equalsIgnoreCase("green"))
    {
        green.push(TShirt);
        sizeInStock[TShirt.getSize()] += 1;
    }
    else
    {
        System.out.println("you shouldnt be here in add");
    }
    /*
    switch (color) {
        case "Red":
            red.push(TShirt);
            sizeInStock[TShirt.getSize()] += 1;
            break;
        case "Blue":
            blue.push(TShirt);
            sizeInStock[TShirt.getSize()] += 1;
            break;
        case "Green":
            green.push(TShirt);
            sizeInStock[TShirt.getSize()] += 1;
            break;
        default:
```

```
            System.out.println("you shouldnt be here in add");
            break;
    }

     */
}

public void sell(String color) {
    if(color.equalsIgnoreCase("red"))
    {
        if (!red.isEmpty())
        {
            TShirts TShirtRed = red.pop();
            sizeInStock[TShirtRed.getSize()] -= 1;
            soldOut[TShirtRed.getSize()] += 1;
            order();
        }
    }
    else if (color.equalsIgnoreCase("blue"))
    {
        if (!blue.isEmpty())
        {
            TShirts TShirtBlue = blue.pop();
            sizeInStock[TShirtBlue.getSize()] -= 1;
            soldOut[TShirtBlue.getSize()] += 1;
            order();
        }

    }
    else if(color.equalsIgnoreCase("green"))
    {
        if (!green.isEmpty())
        {
            TShirts TShirtGreen = green.pop();
            sizeInStock[TShirtGreen.getSize()] -= 1;
            soldOut[TShirtGreen.getSize()] += 1;
            order();
        }

    }
      /*
    switch (color) {
        case "red":
            TShirts TShirtRed = red.pop();
            sizeInStock[TShirtRed.getSize()] -= 1;
            soldOut[TShirtRed.getSize()] += 1;
            order();
            break;
        case "blue":
            TShirts TShirtBlue = blue.pop();
```

```
            sizeInStock[TShirtBlue.getSize()] -= 1;
            soldOut[TShirtBlue.getSize()] += 1;
            order();
            break;
        case "green":
            TShirts TShirtGreen = green.pop();
            sizeInStock[TShirtGreen.getSize()] -= 1;
            soldOut[TShirtGreen.getSize()] += 1;
            order();
            break;
        default:
            //System.out.println("you shouldnt be here in sell");
            break;
    }

        */
}



public void order() {
    //System.out.println("you are in the order method");
    for (int i = 0; i < sizeInStock.length; i++) {
        // System.out.println("you are inside order for loop");
        if (sizeInStock[i] < STOCK_LIMIT) {
            //System.out.println("you are inside order for loop if statement");
            TShirts TShirtRed = new TShirts(i, "red");
            TShirts TShirtBlue = new TShirts(i, "blue");
            TShirts TShirtGreen = new TShirts(i, "green");
            for (int j = 0; j < STOCK_LIMIT; j++) {
                //System.out.println("you are inside the for loop to add the t shirts");
                red.push(TShirtRed);
                blue.push(TShirtBlue);
                green.push(TShirtGreen);
                sizeInStock[i] += 3;
            }
        }
    }
}

private static String colorStackString(Stack<TShirts> stack) {
    final String[] size = {"S", "M", "L", "XL", "Big" };

    String[] strArr = new String[stack.size()];
    Object[] arr = stack.toArray();

    for (int i = 0; i < stack.size(); i++) {
        TShirts ts = (TShirts) arr[i];
        strArr[i] = size[ts.getSize()];
    }
```

```
        return String.join(",", strArr);
    }

    private static String stockSoldString(Integer[] arr) {
        return "S:" + arr[0] + " M:" + arr[1] + " L:" + arr[2] + " XL:" + arr[3] + " Big:" + +arr[4];
    }

    //sell method
    //order method
    //toString method
    @Override
    public String toString() {
        /*
                TShirt Stacks by color
                red=S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
                green=S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
                blue=S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
                sizesInStock=S:9 M:9 L:9 XL:9 Big:9
                soldOut=S:0 M:0 L:0 XL:0 Big:0
        */

        String res = "TShirt Stacks by color\n";

        res += "red=" + colorStackString(red) + "\n";
        res += "green=" + colorStackString(green) + "\n";
        res += "blue=" + colorStackString(blue) + "\n";
        res += "sizesInStock=" + stockSoldString(sizeInStock) + "\n";
        res += "soldOut=" + stockSoldString(soldOut) + "\n";

        return res;
    }

    //main method. Please change this completely
    public static void main(String[] args) {
        String colors[] = {"Red", "Green", "Blue" };
        TShirtStack tsh = new TShirtStack();
        System.out.println(tsh);
        for (int i = 0; i < 5; i++) {
            tsh.addTShirt(new TShirts(i % (1 + TShirts.MAX_SIZE), colors[i % 3]));
            System.out.println(tsh);
        }
        tsh.sell("red");
        System.out.println(tsh);
        tsh.sell("green");
        System.out.println(tsh);
        tsh.sell("blue");
        System.out.println(tsh);

/*
```

```
        TShirtStack actual = new TShirtStack();
        actual.addTShirt(new TShirts(1, "Red"));
        actual.sell("Red");
        System.out.println(actual.toString());


        TShirtStack tsh = new TShirtStack();
        TShirts tsh1 = new TShirts(2,"Red");
        TShirts tsh2 = new TShirts(3,"Blue");
        tsh.addTShirt(tsh1);
        tsh.addTShirt(tsh2);
        System.out.println(tsh);



        TShirtStack tsh = new TShirtStack();
        TShirts tsh1 = new TShirts(2,"Red");
        TShirts tsh2 = new TShirts(3,"Blue");
        tsh.addTShirt(tsh1);
        tsh.addTShirt(tsh2);
        tsh.sell("blue");
        System.out.println(tsh);

         */
    }
}
```

```
package assignment;

import static org.junit.jupiter.api.Assertions.*;
import java.util.*;

class TShirtStackTest {

    @org.junit.jupiter.api.Test
    void addTShirt() {
        TShirtStack tsh = new TShirtStack();
        TShirts tsh1 = new TShirts(2,"Red");
        TShirts tsh2 = new TShirts(3,"Blue");
        tsh.addTShirt(tsh1);
        tsh.addTShirt(tsh2);
        assertNotNull(tsh);
    }

    @org.junit.jupiter.api.Test
    void sell() {
        TShirtStack tsh = new TShirtStack();
        TShirts tsh1 = new TShirts(2,"Red");
        TShirts tsh2 = new TShirts(3,"Blue");
        tsh.addTShirt(tsh1);
        tsh.addTShirt(tsh2);
        tsh.sell("blue");
        tsh.sell("red");
        tsh.sell("green");
```

```
        assertNotNull(tsh);


    }

    @org.junit.jupiter.api.Test
    void order() {
        TShirtStack tsh = new TShirtStack();
        tsh.order();
        assertNotNull(tsh);

    }
}
```

```
package assignment;
import java.util.*;
public class Vehicles {

    private LinkedList<Integer> vehicles;
    private static final Integer maxSpeed = 90;
    private static final Integer minSpeed = 5;

    /** constructors */
    public Vehicles() {
        vehicles = new LinkedList<>();
    }

    /**
     *constructor which takes ArrayList of Integers as its parameter
     *@param arrayListVehicles of ArrayList of Integers
     */
    public Vehicles (ArrayList<Integer> arrayListVehicles) {

        vehicles = new LinkedList<>();
        Collections.sort(arrayListVehicles, Collections.reverseOrder());

        for (int i = 0; i < arrayListVehicles.size(); i++) {

            Integer vehicleSpeed = arrayListVehicles.get(i);
            String infoString = "add " + vehicleSpeed + " to " + i;

            if (vehicleSpeed < minSpeed)
                vehicleSpeed = minSpeed;
            else if (vehicleSpeed > maxSpeed)
                vehicleSpeed = maxSpeed;

            System.out.println(infoString);

            if (vehicles.size() != 0) {
                vehicleSpeed = vehicles.get(i - 1) - 1;

                if (vehicleSpeed < minSpeed)
                    vehicleSpeed = minSpeed;
            }

            vehicles.add(vehicleSpeed);
        }
    }
```

```
    /**
     * toString: displays vehicle list in the form of a String
     *
     * @return String
     */
    @Override
    public String toString() {
        StringJoiner joiner = new StringJoiner(", ", "[", "]");
        for (int i = 0; i < this.vehicles.size(); i++) {
            joiner.add(vehicles.get(i).toString());
        }
        return joiner.toString();
    }

    /**
     * addNewVehicle: Adds a new vehicle to the list based on speed and every other
vehicle changes their speed
     * accordingly
     * @param speed
     * @param position
     */
    public void addNewVehicle(Integer speed, Integer position) {

        if (speed < minSpeed)
            speed = minSpeed;
        else if (speed > maxSpeed)
            speed = maxSpeed;

        if (position > vehicles.size()) {
            position = vehicles.size();
        }

        vehicles.add(position, speed);

        switch (position) {
            case 0:
                for (int i = position + 1; i < vehicles.size(); i++) {
                    Integer speedChange = vehicles.get(i - 1) - 1;
                    if (speedChange < minSpeed)
                        speedChange = minSpeed;
                    vehicles.set(i, speedChange);
                }
                break;
            default:
                if ( position == vehicles.size() - 1 ) {
                    Integer speedChange = vehicles.get(position - 1) - 1;
                    vehicles.set(position, speedChange);
                }
                else {
                    //System.out.println("made it into here");
                    Integer speedChange = vehicles.get(position - 1) - 1;
                    vehicles.set(position, speedChange);
                    for (int i = position + 1; i < vehicles.size(); i++) {
                        speedChange = vehicles.get(i - 1) -1;
                        if (speedChange < minSpeed)
                            speedChange = minSpeed;
                        vehicles.set(i, speedChange);
                    }
                }
```

```
        }

    }

    public static void main (String[] args) {
        Vehicles emptyList = new Vehicles();
        ArrayList<Integer> inputList = new ArrayList<>();
        inputList.add(30);
        inputList.add(99);
        inputList.add(22);
        inputList.add(4);
        inputList.add(25);
        Vehicles fullList = new Vehicles(inputList);
        System.out.println("emptyList: " + emptyList);
        System.out.println("fullList" + fullList);
        fullList.addNewVehicle(20, 0);
        System.out.println("After adding 20 to position 0:\n" + fullList);
        fullList.addNewVehicle(30, 3);
        System.out.println("After adding 30 to position 3:\n" + fullList);
        fullList.addNewVehicle(100, 5);
        System.out.println("After adding 100 to position 5:\n" + fullList);
        fullList.addNewVehicle(70, 8);
        System.out.println("After adding 70 to position 8:\n" + fullList);
        fullList.addNewVehicle(8, 0);
        System.out.println("After adding 8 to position 0:\n" + fullList);
        fullList.addNewVehicle(100000, 0);
            System.out.println("After  adding  ten  million  thousand  to  position  0:\n"  +
fullList);

    }

}
```

```
package assignment;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Assertions;
import java.util.*;


class VehiclesTest<Vehicle> {

    @org.junit.jupiter.api.Test
    @DisplayName("Testing addNewVehicle with a value")
    void addNewVehicle1() {
        Vehicles vehicle = new Vehicles();

        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        Assertions.assertEquals("[30, 29, 28, 27]", vehicle.toString());
    }

    @org.junit.jupiter.api.Test
    @DisplayName("Testing addNewVehicle with a value")
    void addNewVehicle2() {
        Vehicles vehicle = new Vehicles();
```

```
        vehicle.addNewVehicle(90, 5);
        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        vehicle.addNewVehicle(30, 5);
        Assertions.assertEquals("[90, 89, 88, 87, 86]", vehicle.toString());
    }

    @org.junit.jupiter.api.Test
    @DisplayName("Testing addNewVehicle with large values")
    void addNewVehicle3() {
        Vehicles vehicle = new Vehicles();


        vehicle.addNewVehicle(50, 100000);
        vehicle.addNewVehicle(-5, 1);
        Assertions.assertEquals("[50, 49]", vehicle.toString());
    }

    @org.junit.jupiter.api.Test
    @DisplayName("Testing addNewVehicle with large values")
    void addNewVehicle4() {
        Vehicles vehicle = new Vehicles();


        vehicle.addNewVehicle(6, 0);
        vehicle.addNewVehicle(0, 1);
        vehicle.addNewVehicle(0, 1);
        vehicle.addNewVehicle(0, 1);
        Assertions.assertEquals("[6, 5, 5, 5]", vehicle.toString());
    }

}
```

```
package assignment;

public class QA {

    private String qid;
    private String text;
    private String yesQID;
    private String noQID;

    /**
     * Default constructor, initializes QA with default calues
     */
    public QA() {
        this.qid = "";
        this.text = "";
        this.yesQID = "";
        this.noQID = "";
    }

    /**
     * Parameterized Constructor
     * @param qid the question ID
     * @param text the text of the question
     * @param yesQID the question to go to when the user answers yes to this question
     * @param noQID the question to go to when the user answers no to this question
```

```
     */
    public QA(String qid, String text, String yesQID, String noQID) {
        this.qid = qid;
        this.text = text;
        this.yesQID = yesQID;
        this.noQID = noQID;
    }

    /**
     * A constructor that parses a string representation of the question
      * @param qString a string in the format of qid,text,yesQID,noQID (e.g. Q1,Is it
big?,Q2,Q5)
     */
    public QA(String qString) {
        String[] parts = qString.split(",", 4);
        if (parts.length != 4) {
            return;
        }

        this.qid = parts[0];
        this.text = parts[1];
        this.yesQID = parts[2];
        this.noQID = parts[3];
    }

    public String getQid() {
        return qid;
    }

    public void setQid(String qid) {
        this.qid = qid;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public String getYesQID() {
        return yesQID;
    }

    public void setYesQID(String yesQID) {
        this.yesQID = yesQID;
    }

    public String getNoQID() {
        return noQID;
    }

    public void setNoQID(String noQID) {
        this.noQID = noQID;
    }

    @Override
    public String toString() {
```

```
        return "qid=" + getQid() + "\ntext=" + getText() + "\nYesQID=" + getYesQID() +
"\nNoQID=" + getNoQID() + "\n";
    }

    public static void main(String[] args) {

    }
}


package assignment;

import java.util.Hashtable;

public class QuizHashTable {
    private Hashtable<String, QA> quiz;

    /**
     * Constructs a QuizHashTable from a string of questions
     *
     * @param quizStr A string of "-" seperated questions
     */
    public QuizHashTable(String quizStr) {


        this.quiz = new Hashtable<>();
        String[] questions = quizStr.split("-");

        for (String question : questions) {
            QA qa = new QA(question);
            quiz.put(qa.getQid(), qa);
        }
    }

    public String ask(String qid, String answerSequence) {
        String res = new String();
        QA currentQuestion = quiz.get(qid);
        for (Character answer : answerSequence.toCharArray()) {
            res += currentQuestion.getText();

            if (answer == 'Y') {
                res += answer;
                currentQuestion = quiz.get(currentQuestion.getYesQID());
            } else if (answer == 'N') {
                res += answer;
                currentQuestion = quiz.get(currentQuestion.getNoQID());
            }

            res += "\n";
        }

        res += currentQuestion.getText();

        return res;
    }

    @Override
    public String toString() {
        String res = new String();
```

```
        for ( QA qa : quiz.values()) {
            res += qa.toString();
        }

        return res;
    }

    public static void main(String[] args) {
        //All questions are given in rawQ String
            final String rawQ = "Q1,Is it big?,Q2,Q5-Q2,Is it white?,Q3,Q4-Q3,Whale,
,-Q4,Cat, ,-Q5,Is it an animal?,Q6,Q7-Q6,Ant, ,-Q7,Dust, ,";
        //Possible answers from the user
        final String ans1 = "YY";
        final String ans2 = "YN";
        final String ans3 = "NY";
        final String ans4 = "NN";
        //First get all questions and in the constructor split them.
        //Use - as the separator.
        //Store the parts that correspond to single questions in the array
          // For each question(array element) call the constructor of QA and create a QA
object.
         // QA constructor gets a String parameter and splits it into parts by using , as
the separator
          //Insert these QA objects into Hashtable quiz by using the qid field as the key
and QA object as the value
        QuizHashTable qz = new QuizHashTable(rawQ);
        //res is the output you will generate and return from the ask method
        String res;
        // ask method gets a question id to start and an answer sequence
          //For the following start from question whose id is Q1 and answer sequence is
ans1 i.e. YY
        res = qz.ask("Q1", ans1);
        System.out.println(res);
        // Q1 is Is it big?
        // and first answer is Y
        //So ask the question Is it an animal?
        //and the second answer is Y
          //Since there is no more question stop and give the answer as the text of the
current QA object whichis Ant
        //Output res will be
        /*
        Is it big?Y
        Is it an animal?Y
        Ant
        */
        res = qz.ask("Q1", ans2);
        System.out.println(res);
         // See the explanations above for details. This time we start again from Q1 but
answer sequence ans2 is YN
        //Output is
        /*
        Is it big?Y
        Is it white?N
        Cat
        */
        res = qz.ask("Q1", ans3);
        System.out.println(res);
         // See the explanations above for details. This time we start again from Q1 but
answer sequence ans3 is NY
```

```
        //Output is
        /*
        Is it big?N
        Is it an animal?Y
        Ant
        */
        res = qz.ask("Q1", ans4);
        System.out.println(res);
         // See the explanations above for details. This time we start again from Q1 but
answer sequence ans4is NN
        //Output is
        /*
        Is it big?N
        Is it an animal?N
        Dust
        */

    }
}

package assignment;

import static org.junit.Assert.*;

public class QATest {

    @org.junit.Test
    public void getSetQid() {
        QA qa = new QA();
        qa.setQid("Q1");
        assertEquals("Q1", qa.getQid());
    }

    @org.junit.Test
    public void getSetText() {
        QA qa = new QA();
        qa.setText("text");
        assertEquals("text", qa.getText());
    }

    @org.junit.Test
    public void getSetYesQID() {
        QA qa = new QA();
        qa.setYesQID("Q1");
        assertEquals("Q1", qa.getYesQID());
    }

    @org.junit.Test
    public void getSetNoQID() {
        QA qa = new QA();
        qa.setNoQID("Q1");
        assertEquals("Q1", qa.getNoQID());
    }

    @org.junit.Test
    public void testParameterizedConstructor() {
        QA qa = new QA("Q1", "text", "yes", "no");
        assertEquals("Q1", qa.getQid());
        assertEquals("text", qa.getText());
```

```
        assertEquals("yes", qa.getYesQID());
        assertEquals("no", qa.getNoQID());
    }

    @org.junit.Test
    public void testStringConstructor() {
        QA qa = new QA("Q1,text,yes,no");
        assertEquals("Q1", qa.getQid());
        assertEquals("text", qa.getText());
        assertEquals("yes", qa.getYesQID());
        assertEquals("no", qa.getNoQID());
    }

    @org.junit.Test
    public void main() {
    }
}

package assignment;

import org.junit.Test;

import java.util.StringJoiner;

import static org.junit.Assert.*;

public class QuizHashTableTest {

    @Test
    public void ask() {
        StringJoiner quizStr = new StringJoiner("-");
        quizStr.add("Q1,Enter the tavern?,Q3,Q2");
        quizStr.add("Q2,You were eaten by wolves,,");
        quizStr.add("Q3,Sit down at the table?,Q4,Q5");
        quizStr.add("Q4,You sat on spikes,,");
        quizStr.add("Q5,Go to the counter?,Q7,Q6");
        quizStr.add("Q6,You fall down the trap door underneath you,,");
        quizStr.add("Q7,You win!!,,");

        QuizHashTable quiz = new QuizHashTable(quizStr.toString());

        String wolvesAns = "N";
        String wolvesExpected = "Enter the tavern?N\nYou were eaten by wolves";

        String spikesAns = "YY";
         String spikesExpected = "Enter the tavern?Y\nSit down at the table?Y\nYou sat on
spikes";

        String trapdoorAns = "YNN";
         String trapdoorExpected = "Enter the tavern?Y\nSit down at the table?N\nGo to the
counter?N\nYou fall down the trap door underneath you";

        String winAns = "YNY";
          String winExpected = "Enter the tavern?Y\nSit down at the table?N\nGo to the
counter?Y\nYou win!!";

        assertEquals(wolvesExpected, quiz.ask("Q1", wolvesAns));
        assertEquals(spikesExpected, quiz.ask("Q1", spikesAns));
        assertEquals(trapdoorExpected, quiz.ask("Q1", trapdoorAns));
```

```
        assertEquals(winExpected, quiz.ask("Q1", winAns));
    }
}
```

B. Sample output 1

I. Describe your test 1:

Tested out the TShirts and TShirtsStack classes in main. Output matched the output given in the assignment pdf.

II. Text output 1:

TShirt Stacks by color
red=
green=
blue=
sizesInStock=S:0 M:0 L:0 XL:0 Big:0
soldOut=S:0 M:0 L:0 XL:0 Big:0

TShirt Stacks by color
red=S
green=
blue=
sizesInStock=S:1 M:0 L:0 XL:0 Big:0
soldOut=S:0 M:0 L:0 XL:0 Big:0

TShirt Stacks by color
red=S
green=M
blue=
sizesInStock=S:1 M:1 L:0 XL:0 Big:0
soldOut=S:0 M:0 L:0 XL:0 Big:0

TShirt Stacks by color
red=S
green=M
blue=L
sizesInStock=S:1 M:1 L:1 XL:0 Big:0
soldOut=S:0 M:0 L:0 XL:0 Big:0

TShirt Stacks by color
red=S,XL
green=M
blue=L
sizesInStock=S:1 M:1 L:1 XL:1 Big:0
soldOut=S:0 M:0 L:0 XL:0 Big:0

TShirt Stacks by color
red=S,XL
green=M,Big
blue=L
sizesInStock=S:1 M:1 L:1 XL:1 Big:1

soldOut=S:0 M:0 L:0 XL:0 Big:0

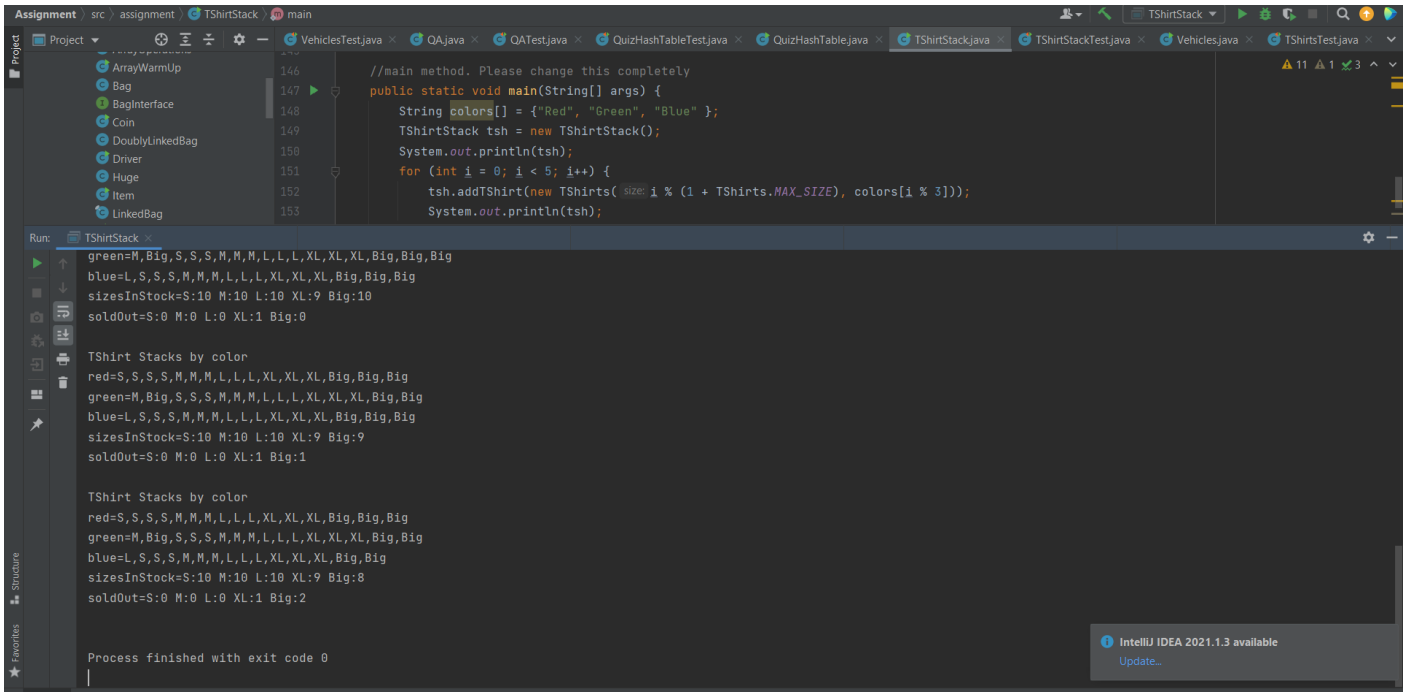
TShirt Stacks by color
red=S,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
green=M,Big,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
blue=L,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
sizesInStock=S:10 M:10 L:10 XL:9 Big:10
soldOut=S:0 M:0 L:0 XL:1 Big:0

TShirt Stacks by color
red=S,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
green=M,Big,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big
blue=L,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
sizesInStock=S:10 M:10 L:10 XL:9 Big:9
soldOut=S:0 M:0 L:0 XL:1 Big:1

TShirt Stacks by color
red=S,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big,Big
green=M,Big,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big
blue=L,S,S,S,M,M,M,L,L,L,XL,XL,XL,Big,Big
sizesInStock=S:10 M:10 L:10 XL:9 Big:8
soldOut=S:0 M:0 L:0 XL:1 Big:2

Process finished with exit code 0

III. Screenshot 1:



C. Sample output 2

I. Describe your test 2:

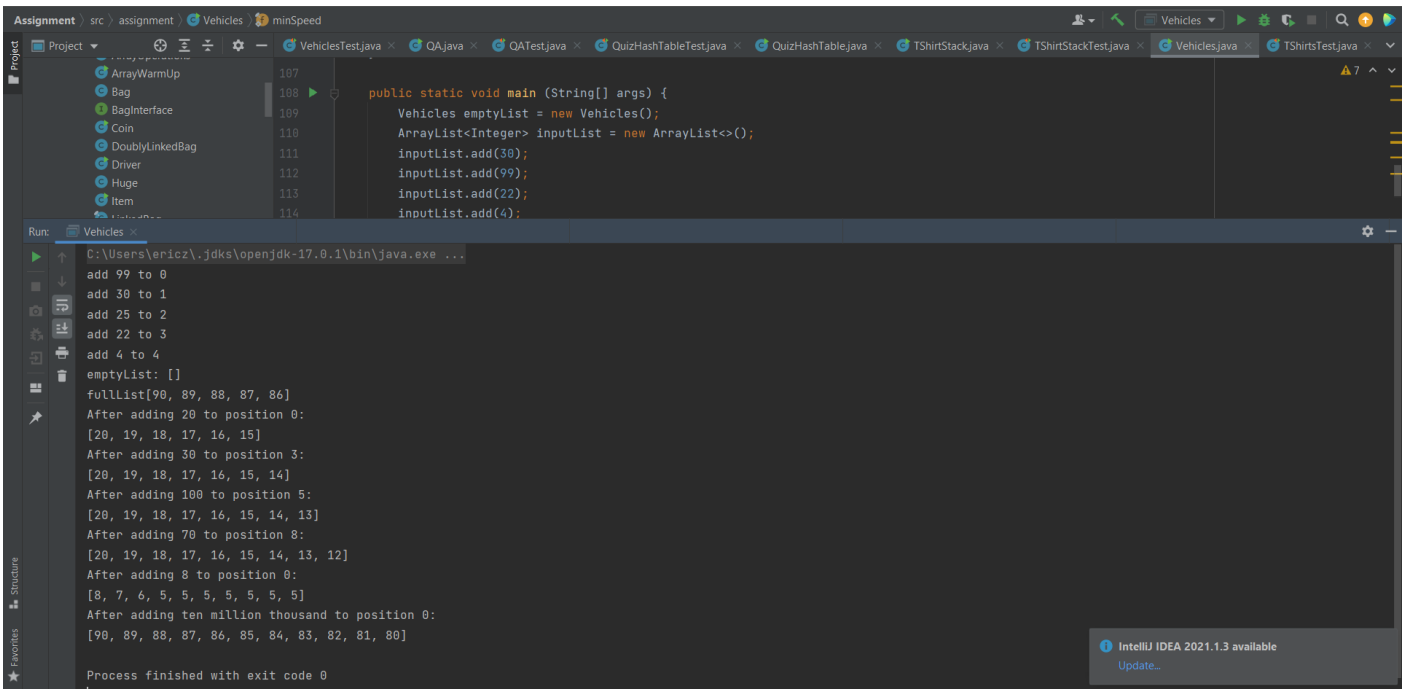
Tested out Vehicles class. Output shows the class functions as expected. The speeds of the cars always follow the rules.

II. Text output 2:

```
add 99 to 0
add 30 to 1
add 25 to 2
add 22 to 3
add 4 to 4
emptyList: []
fullList[90, 89, 88, 87, 86]
After adding 20 to position 0:
[20, 19, 18, 17, 16, 15]
After adding 30 to position 3:
[20, 19, 18, 17, 16, 15, 14]
After adding 100 to position 5:
[20, 19, 18, 17, 16, 15, 14, 13]
After adding 70 to position 8:
[20, 19, 18, 17, 16, 15, 14, 13, 12]
After adding 8 to position 0:
[8, 7, 6, 5, 5, 5, 5, 5, 5, 5]
After adding ten million thousand to position 0:
[90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80]

Process finished with exit code 0
```

III. Screenshot 2:



D. Sample output 3

I. Describe your test 3:

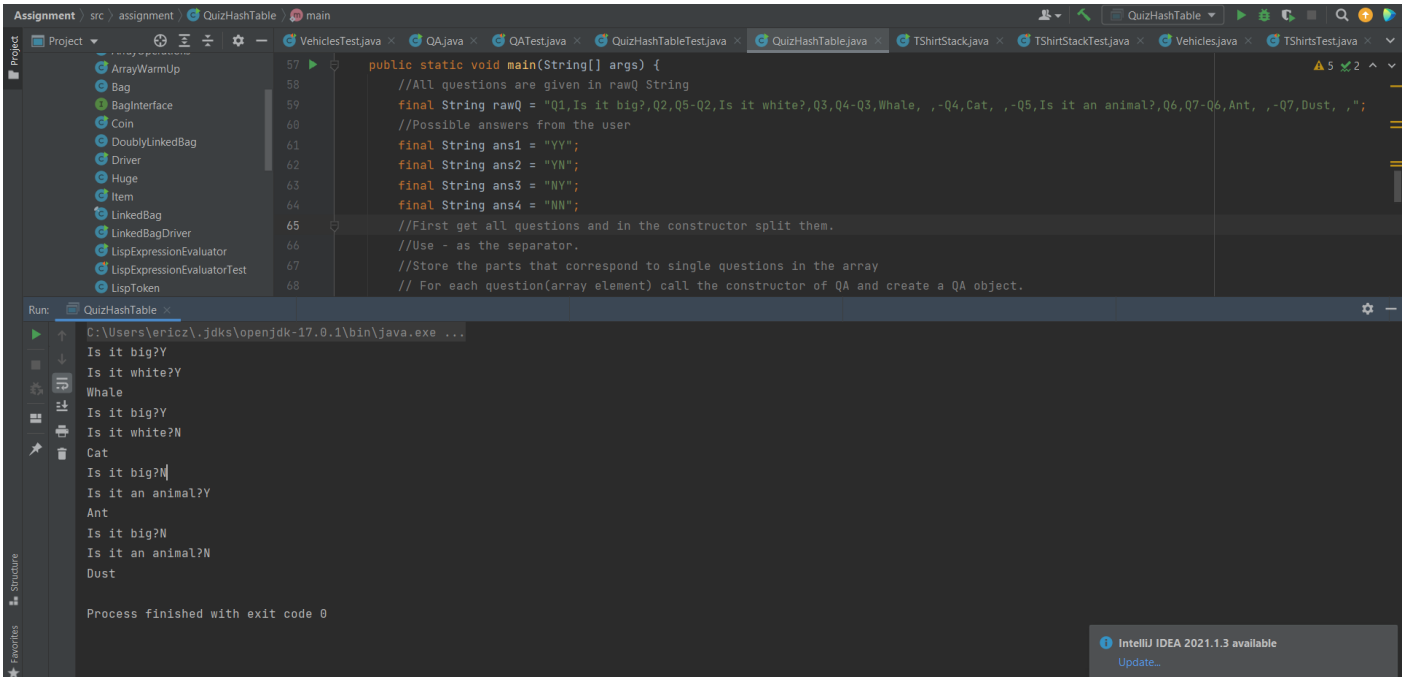
Tested out the QuizHashTable class with various responses to the questions. The appropriate answers/next question are given after each round of questioning.

II. Text output 3:

Is it big?Y
Is it white?Y
Whale
Is it big?Y
Is it white?N
Cat
Is it big?N
Is it an animal?Y
Ant
Is it big?N
Is it an animal?N
Dust

Process finished with exit code 0

III. Screenshot 3:


Assignment
src
assignment
QuizHashTable
main
QuizHashTable
Project
VehiclesTest.java
QA.java
QATest.java
QuizHashTableTest.java
QuizHashTable.java
TShirtStack.java
TShirtStackTest.java
Vehicles.java
TShirtsTest.java
ArrayWarmUp
Bag
BagInterface
Coin
DoublyLinkedBag
Driver
Huge
Item
LinkedBag
LinkedBagDriver
LispExpressionEvaluator
LispExpressionEvaluatorTest
LispToken
public static void main(String[] args) {
//All questions are given in rawQ String
final String rawQ = "Q1,Is it big?,Q2,Q5-Q2,Is it white?,Q3,Q4-Q3,Whale, ,-Q4,Cat, ,-Q5,Is it an animal?,Q6,Q7-Q6,Ant, ,-Q7,Dust, ,";
//Possible answers from the user
final String ans1 = "YY";
final String ans2 = "YN";
final String ans3 = "NY";
final String ans4 = "NN";
//First get all questions and in the constructor split them.
//Use - as the separator.
//Store the parts that correspond to single questions in the array
// For each question(array element) call the constructor of QA and create a QA object.
Run:
QuizHashTable
C:\Users\ericz\.jdks\openjdk-17.0.1\bin\java.exe ...
Is it big?Y
Is it white?Y
Whale
Is it big?Y
Is it white?N
Cat
Is it big?N
Is it an animal?Y
Ant
Is it big?N
Is it an animal?N
Dust
Process finished with exit code 0
IntelliJ IDEA 2021.1.3 available
Update...
Structure
Favorites