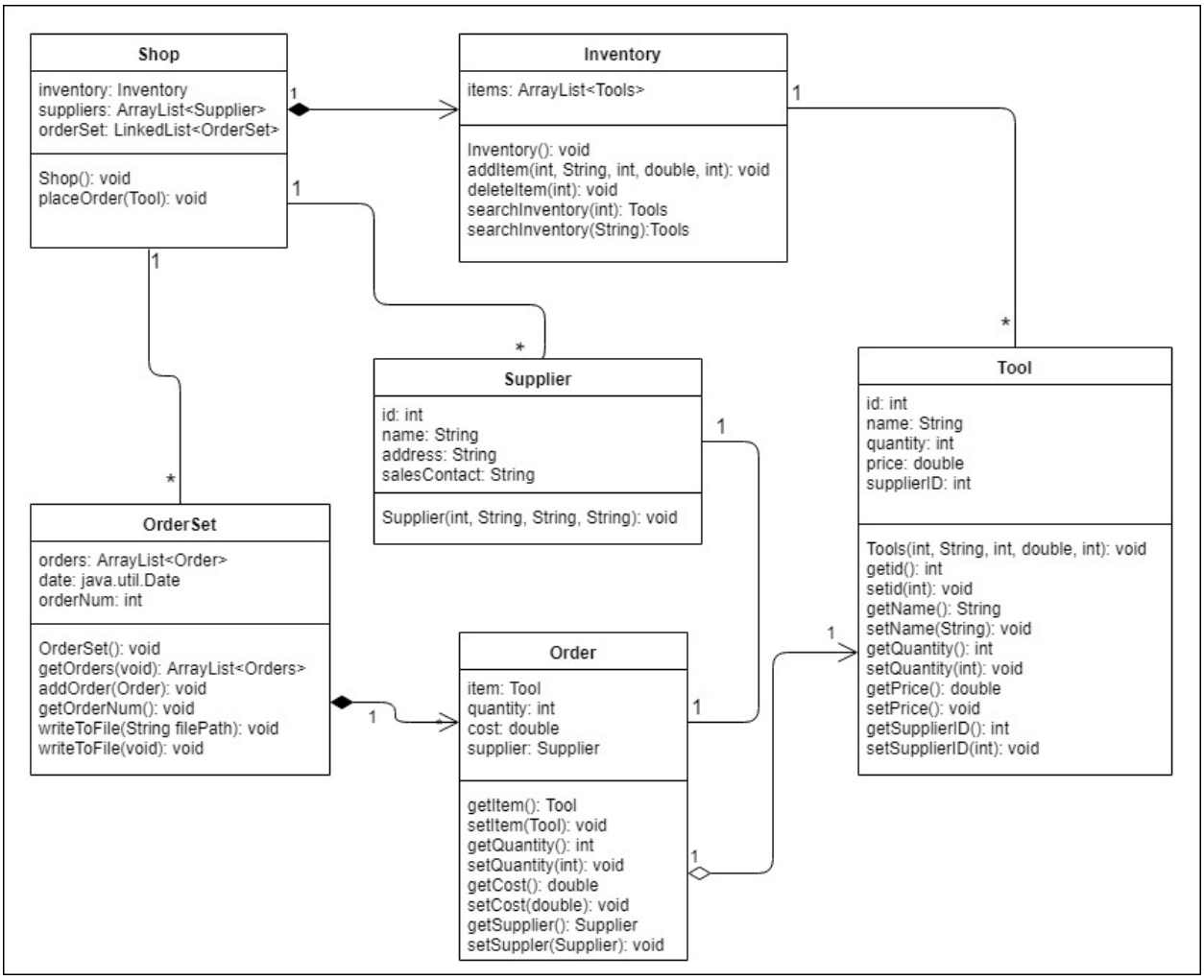
Course: ENSF 409

Student Name: Antonio Santos

Lab number: Lab 2

Exercise 3


Shop
inventory: Inventory
suppliers: ArrayList<Supplier>
orderSet: LinkedList<OrderSet>
Shop(): void
placeOrder(Tool): void
Inventory
items: ArrayList<Tools>
Inventory(): void
addItem(int, String, int, double, int): void
deleteItem(int): void
searchInventory(int): Tools
searchInventory(String):Tools
Supplier
id: int
name: String
address: String
salesContact: String
Supplier(int, String, String, String): void
Tool
id: int
name: String
quantity: int
price: double
supplierID: int
Tools(int, String, int, double, int): void
getid(): int
setid(int): void
getName(): String
setName(String): void
getQuantity(): int
setQuantity(int): void
getPrice(): double
setPrice(): void
getSupplierID(): int
setSupplierID(int): void
OrderSet
orders: ArrayList<Order>
date: java.util.Date
orderNum: int
OrderSet(): void
getOrders(void): ArrayList<Orders>
addOrder(Order): void
getOrderNum(): void
writeToFile(String filePath): void
writeToFile(void): void
Order
item: Tool
quantity: int
cost: double
supplier: Supplier
getItem(): Tool
setItem(Tool): void
getQuantity(): int
setQuantity(int): void
getCost(): double
setCost(double): void
getSupplier(): Supplier
setSuppler(Supplier): void
1
*

Exercise 4

```java
import java.io.*;

//STUDENTS SHOULD ADD CLASS COMMENTS, METHOD COMMENTS, FIELD COMMENTS

/**
 * Provides data fields and methods to create a tic-tac-toe game
 * in a Java application
 * This class is the starting point of the program
 *
 * @author Antonio Santos
 * @version 1.0
 * @since January 31, 2019
*/
public class Game implements Constants {
    /**
     * The board that will be used in the game
     */
    private Board theBoard;
    /**
     * The moderator that will be in charge of watching the game
     */
    private Referee theRef;

    /**
     * Creates a new Board object to play on
     */
    public Game( ) {
        theBoard  = new Board();
    }

    /**
     * Sets the referee and tells it to start the game
     * @param r sets the referee that will be running the game
     * @throws IOException
     */
    public void appointReferee(Referee r) throws IOException {
        theRef = r;
        theRef.runTheGame();
    }

    public static void main(String[] args) throws IOException {
        Referee theRef;
```

```
        Player xPlayer, oPlayer;
        BufferedReader stdin;
        Game theGame = new Game();
        stdin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("\nPlease enter the name of the \'X\' player: ");
        String name= stdin.readLine();
        while (name == null) {
            System.out.print("Please try again: ");
            name = stdin.readLine();
        }

        xPlayer = new Player(name, LETTER_X);
        xPlayer.setBoard(theGame.theBoard);

        System.out.print("\nPlease enter the name of the \'O\' player: ");
        name = stdin.readLine();
        while (name == null) {
            System.out.print("Please try again: ");
            name = stdin.readLine();
        }

        oPlayer = new Player(name, LETTER_O);
        oPlayer.setBoard(theGame.theBoard);

        theRef = new Referee();
        theRef.setBoard(theGame.theBoard);
        theRef.setoPlayer(oPlayer);
        theRef.setxPlayer(xPlayer);

        theGame.appointReferee(theRef);
    }


}
```

```
public interface Constants {
    static final char SPACE_CHAR = ' ';
    static final char LETTER_O = 'O';
    static final char LETTER_X = 'X';
}
```

```java
//STUDENTS SHOULD ADD CLASS COMMENTS, METHOD COMMENTS, FIELD COMMENTS

/**
 * This class is in charge of displaying the board on the console
 * and adding the marks that the players put on the board.
 * It will also be in charge of checking for a winner or if the
 * board is full.
 *
 * @author Antonio Santos
 * @version 1.0
 * @since January 31, 2019
 */
public class Board implements Constants {
    /**
     * The board that will be displayed on the console
     */
    private char theBoard[][];
    /**
     * the number of marks on the board
     */
    private int markCount;

    /**
     * creates the the board matrix.
     */
    public Board() {
        markCount = 0;
        theBoard = new char[3][];
        for (int i = 0; i < 3; i++) {
            theBoard[i] = new char[3];
            for (int j = 0; j < 3; j++)
                theBoard[i][j] = SPACE_CHAR;
        }
    }

    /**
     * Returns the mark placed in the wanted element
     * @param row   index of row
     * @param col   index of column
     * @return  the character within the wanted element
     */
    public char getMark(int row, int col) {
        return theBoard[row][col];
```

```
    }

    /**
     * checks if the board is full
     * @return  the markCount = to 9;
     */
    public boolean isFull() {
        return markCount == 9;
    }

    /**
     * will constantly check the board to see if xPlayer has won the game
     * @return returns whether the player has won or not
     */
    public boolean xWins() {
        if (checkWinner(LETTER_X) == 1)
            return true;
        else
            return false;
    }

    /**
     * will constantly check the board to see if xPlayer has won the game
     * @return returns whether the player has won or not
     */
    public boolean oWins() {
        if (checkWinner(LETTER_O) == 1)
            return true;
        else
            return false;
    }

    /**
     * displays the board on the console.
     */
    public void display() {
        displayColumnHeaders();
        addHyphens();
        for (int row = 0; row < 3; row++) {
            addSpaces();
            System.out.print("    row " + row + ' ');
            for (int col = 0; col < 3; col++)
                System.out.print("|  " + getMark(row, col) + "  ");
            System.out.println("|");
            addSpaces();
```

```
            addHyphens();
        }
    }

    /**
     * Sets the element with the mark given by the player and
     * increases the mark count
     * @param row   index of row
     * @param col   index of col
     * @param mark  mark that will be placed in the element
     */
    public void addMark(int row, int col, char mark) {

        theBoard[row][col] = mark;
        markCount++;
    }

    /**
     * clears the board and resets the mark count
     */
    public void clear() {
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                theBoard[i][j] = SPACE_CHAR;
        markCount = 0;
    }

    /**
     * checks if a player has won the game
     * @param mark  the mark that will be used to check for winner
     * @return returns result. will return 1 if there is a winner else 0
     */
    int checkWinner(char mark) {
        int row, col;
        int result = 0;

        for (row = 0; result == 0 && row < 3; row++) {
            int row_result = 1;
            for (col = 0; row_result == 1 && col < 3; col++)
                if (theBoard[row][col] != mark)
                    row_result = 0;
            if (row_result != 0)
                result = 1;
        }
```

```
        for (col = 0; result == 0 && col < 3; col++) {
            int col_result = 1;
            for (row = 0; col_result != 0 && row < 3; row++)
                if (theBoard[row][col] != mark)
                    col_result = 0;
            if (col_result != 0)
                result = 1;
        }

        if (result == 0) {
            int diag1Result = 1;
            for (row = 0; diag1Result != 0 && row < 3; row++)
                if (theBoard[row][row] != mark)
                    diag1Result = 0;
            if (diag1Result != 0)
                result = 1;
        }
        if (result == 0) {
            int diag2Result = 1;
            for (row = 0; diag2Result != 0 && row < 3; row++)
                if (theBoard[row][3 - 1 - row] != mark)
                    diag2Result = 0;
            if (diag2Result != 0)
                result = 1;
        }
        return result;
    }

    /**
     * displays the column header
     */
    void displayColumnHeaders() {
        System.out.print("          ");
        for (int j = 0; j < 3; j++)
            System.out.print("|col " + j);
        System.out.println();
    }

    /**
     * displays the hyphens
     */
    void addHyphens() {
        System.out.print("          ");
        for (int j = 0; j < 3; j++)
```

```
            System.out.print("+-----");
        System.out.println("+");
    }
    /**
     * adds white space
     */
    void addSpaces() {
        System.out.print("          ");
        for (int j = 0; j < 3; j++)
            System.out.print("|     ");
        System.out.println("|");
    }
}
```

```
/**
 * This class will be in charge of setting the board and setting the players.
 *
 * @author Antonio Santos
 * @version 1.0
 * @since January 31, 2019
 */
public class Referee{
    /**
     * Player that uses the X mark
     */
    private Player xPlayer;
    /**
     * Player that uses the O mark
     */
    private Player oPlayer;
    /**
     * The board that will be used in the game.
     */
    private Board board;

    /**
     * Constructor for referee
     */
    public Referee(){}

    /**
     * Begins the game and sets the players as opponents
```

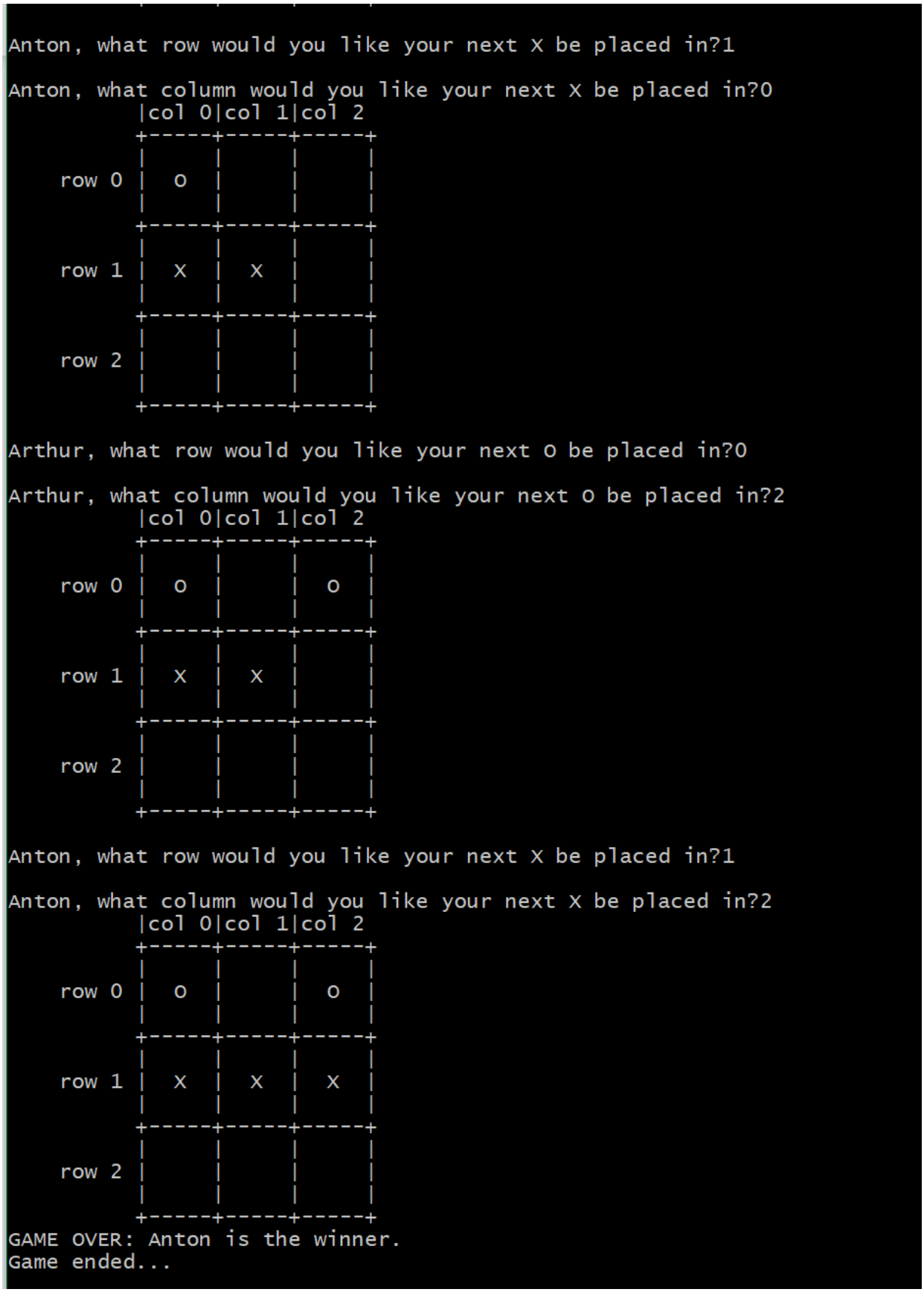
```
     * of one another
     */
    public void runTheGame(){
        xPlayer.setOpponent(oPlayer);
        oPlayer.setOpponent(xPlayer);
        board.display();
        xPlayer.play();
    }

    /**
     * sets the game board
     * @param board the game board
     */
    public void setBoard(Board board){
        this.board = board;
    }
    /**
     * sets the player that will use the O mark
     * @param oPlayer Player that will use the O mark
     */
    public void setoPlayer(Player oPlayer){
        this.oPlayer = oPlayer;
    }
    /**
     * sets the player that will use the X mark
     * @param xPlayer Player that will use the X mark
     */
    public void setxPlayer(Player xPlayer){
        this.xPlayer = xPlayer;
    }
}
```

```
import java.util.Scanner;

/**
 * This class takes care of all of the information about the player
 * and also where the game moves are executed.
 *
 * @author Antonio Santos
 * @version 1.0
 * @since January 31, 2019
 */
public class Player implements Constants{
    /**
```

```java
     * name of the player
     */
    private String name;
    /**
     * the game board that will be used in the game
     */
    private Board board;
    /**
     * the opponent of the current player
     */
    private Player opponent;
    /**
     * the mark of the player
     */
    private char mark;

    /**
     * Player constructor sets the name and mark
     * @param name name of the player
     * @param mark mark to be used by the player
     */
    public Player(String name, char mark){
        this.name = name;
        this.mark = mark;
    }

    /**
     * Where the game starts. Prompts players to make moves and checks
     * every turn if a player has won.
     */
    public void play(){
        while(true){
            makeMove();
            if(board.xWins() == true){
                board.display();
                System.out.println("GAME OVER: " + name + " is the winner.");
                break;
            }else if(board.oWins() == true){
                board.display();
                System.out.println("GAME OVER: " + name + " is the winner.");
                break;
            }else if(board.isFull() == true){
                board.display();
                System.out.println("GAME OVER: game ended in a tie");
                break;
```

```
            }
            board.display();
            opponent.play();
        }
        System.out.println("Game ended...");
        System.exit(1);
    }

    /**
     * Where the players make their moves.
     */
    public void makeMove(){
        System.out.print("\n" + name + ", what row would you like your next " +
mark + " be placed in?");
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        System.out.print("\n" + name + ", what column would you like your next "
+ mark + " be placed in?");
        int col = scan.nextInt();
        if(board.getMark(row, col) == LETTER_O || board.getMark(row, col) ==
LETTER_X){
            System.out.println("\nSpot already taken, please choose an open
space.");
            makeMove();
        }
        board.addMark(row, col, mark);

    }

    /**
     * sets the opponent of a player
     * @param player opponent of current player
     */
    public void setOpponent(Player player){
        opponent = player;
    }

    /**
     * sets board of the game
     * @param board board to be used by the game
     */
    public void setBoard(Board board){
        this.board = board;
    }
}
```

```
Please enter the name of the 'X' player: Anton

Please enter the name of the 'O' player: Arthur
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+

Anton, what row would you like your next X be placed in? 1

Anton, what column would you like your next X be placed in? 1
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |  X  |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+

Arthur, what row would you like your next O be placed in? 0

Arthur, what column would you like your next O be placed in? 0
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |  O  |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |  X  |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
```


Anton, what row would you like your next X be placed in?1
Anton, what column would you like your next X be placed in?0
|col 0|col 1|col 2
row 0 | O |
row 1 | X | X |
row 2 |
Arthur, what row would you like your next O be placed in?0
Arthur, what column would you like your next O be placed in?2
|col 0|col 1|col 2
row 0 | O | | O |
row 1 | X | X |
row 2 |
Anton, what row would you like your next X be placed in?1
Anton, what column would you like your next X be placed in?2
|col 0|col 1|col 2
row 0 | O | | O |
row 1 | X | X | X |
row 2 |
GAME OVER: Anton is the winner.
Game ended...