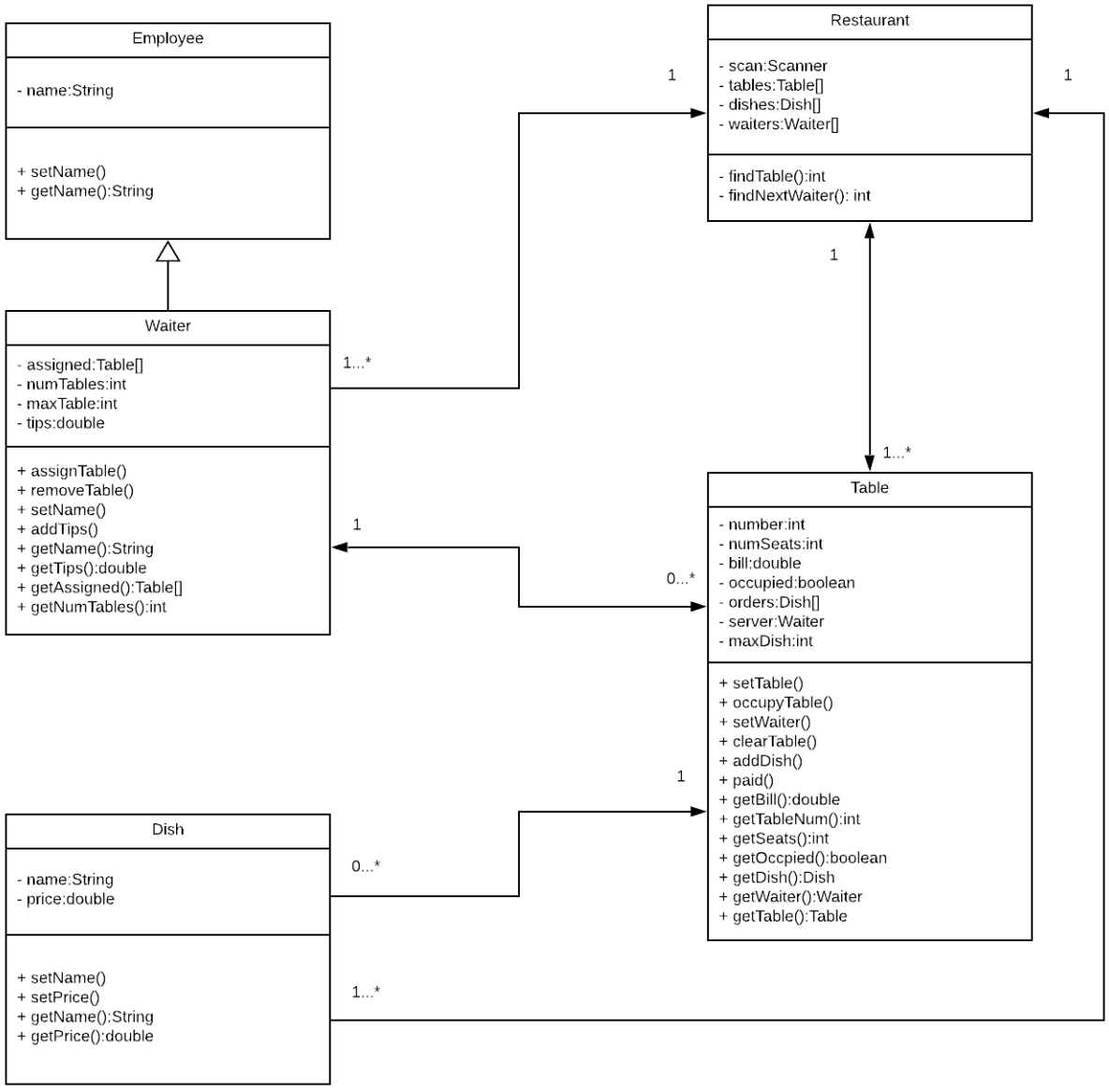
Criterion C: Development

UML

UML Diagram Restaurant Managing Helpful Software



Complex Code

1. Try & Catch handling
    a. Try catch exceptions are used to ensure that users are entering integers for their main menu selections. If an integer is not entered, the system will handle it gracefully and request an integer.

```
System.out.println("Great! The system is now live.");
while(entry != 0){
    System.out.println("0) Exit system.");
    System.out.println("1) Seat a customer.");
    System.out.println("2) View waiter assignments.");
    System.out.println("3) Enter an order.");
    System.out.println("4) View orders.");
    System.out.println("5) Calculate a bill.");
    System.out.println("6) Enter payment.");
    System.out.println("7) Reset a table for the next guest.");
    System.out.println("8) Show restaurant revenue.");
    try{
        entry = scan.nextInt();
    }
    catch(InputMismatchException e){
        System.out.println("Please enter an integer.");
        scan.nextLine();
        entry = -1;
    }
```

2. Nested loops
    a. Nested loops are especially helpful when traversing 2D arrays and other situations with more than one variable to consider. When cycling through tables and their orders, a nested loop is helpful.

```
System.out.println("Here are the waiter's assignments.");
for(int i = 0; i < waiters.length; i ++){
    if (waiters[i] != null){
        System.out.println(waiters[i].getName() + ":");
        for(int j = 0; j < waiters[i].getAssigned().length; j ++){
            if(waiters[i].getAssigned()[j] != null){
                System.out.println("Table " + waiters[i].getAssigned()[j].getTableNum());
            }
        }
        System.out.println();
    }
}
```

3. Sorting
    a. The tables array is sorted with selection sort, which selects the lowest element to

place at the beginning, and then the next lowest to place next, and etcetera. The selection sort helps my system automatically seat guests by party size.

```
//Sorting tables by number of seats
int holdIndex = 0;
int index = 0;
int pointer = 0;
int c = tables.length;
Table hold = null;
Table min = null;
while (index < c){
    pointer = index;
    min = tables[pointer];
    holdIndex = pointer;
    while (pointer < c){
        if (tables[pointer].getSeats() < min.getSeats()){
            min = tables[pointer];
            holdIndex = pointer;
        }
        pointer = pointer + 1;
    }
    hold = tables[index]; // Swap elements after selecting lowest element
    tables[index] = min;
    tables[holdIndex] = hold;
    index = index + 1;
}
```

4. Inheritance
    a. Inheritance helps with code reuse, as classes with different purposes sharing similar variables and methods can now share them by inheriting them from a parent class. Having the Waiter class inherit the Employee class means that a future class of Chefs and Hosts can be coded, also sharing variables with the Waiter class.

```
public class Waiter extends Employee{
    private Table[] assigned;
    private int numTables;
    private final int maxTable = 10;
    private double tips;
```

5. 2D Array
    a. 2D arrays can be used to keep track of many aspects, including which Waiter has which Table as an assignment. By stacking the tables by waiter and by order of arrival, the 2D array helps organize the entire assignment of tables.

```
System.out.println("Next, let's take care of our Waiters.");
System.out.println("How many servers are employed?");
int numWaiters = scan.nextInt();
waiters = new Waiter[numWaiters];
String clear = scan.nextLine();
for(int i = 1; i <= numWaiters; i ++){
    System.out.println("Please enter waiter " + i + "'s name.");
    String input = scan.nextLine();
    waiters[i - 1] = new Waiter(input);
}

Table[][] assignments = new Table[numWaiters][numTables];
```

6. Linear search
    a. A linear search is an easy way to find items in a relatively small array. When arrays get larger to search, a binary search function would be more efficient, but with smaller arrays such as Waiter assignments, a linear search would suffice.

```
private static int findTable(){
    System.out.println("Please select a table number.");
    int input = scan.nextInt();
    for(int i = 0; i < tables.length; i ++){
        if (tables[i] != null && tables[i].getTableNum() == input)
            return i;
    }
    return -1;
}
```