Clarissa Kuter

CSC400 Data Structure and Algorithms

Shaher Daoud

26 August 2022

Online Food Delivery Service Pseudocode

```
//below is the bag interface that will be implemented by the program
public interface BagInterface<T> {
   public int getCurrentSize();
                  //this method will retrieve the current number of entries in the bag
   public boolean add(T newEntry); //this method will allow users to add a new entry to the bag
   public T[] toArray(); //this method will allow users to retrieve all the entries that are in the bag
}
```

```
//below is the class that is used to define and create the FoodItem objects that will be ordered through the delivery service
public class FoodItem {
        private String name;
        private double price;

        //constructor class for the object
        public FoodItem (String name, double price) {
                this.name = name;
                this.price = price;
        }
```

```
	//the get moethods
	public String getName() {
	   return this.name;
	}

	public double getPrice() {
	   return this.price;
	}

	@Override
	public String toString() {
	return this.name + " " + this.price;
	}
}
```

```
//below is the class that will be implementing the BagInterface
public final class FoodBag<T> implements BagInterface<T> {
	private final T[] bag;
	private int numberOfEntries;
	private static final int DEFAULT_CAPACITY = 25;

	public FoodBag() {
		this(DEFAULT_CAPACITY);
	}

	public FoodBag(int capacity) {
		@SuppressWarnings("unchecked")
```

```
            T[] foodBag = (T[]) new Object[DEFAULT_CAPACITY];
            bag = tempBag;
            numberOfEntries = 0;
        }
        //below are what will override the bag interface's initial methods
        @Override
        public boolean add(T newEntry) {
           //will check if the user's bag is full, and if not, this will add a new entry to their bag.
        }
        @Override
        public T[] toArray() {
           //will retrieve all the items in the bag.
        }
        @Override
        public int getCurrentSize() {
           //will retrieve current size of the bag
                return 0;
        }
        private boolean isArrayFull() {
           //used to check if array is full
        }
}
```

```
//below is the demo class that executes the bag
public class BagDemo {
        public static void main(String[] args) {
                BagInterface<FoodItem> adtBag = new FoodBag<>();
```

```
        /*below code declares and initializes the fixed food item array.
         *It determines how many items will fit in the bag.
         *Once I decide how many items I want in the array, I will change x to that
        *number.
         *I will then repeat Food[0] = new FoodItem(" ", $.$$) for x amount of times.
         *" " and $.$$ will also be changed to the actual name and price of each item */

        FoodItem[] Food = new FoodItem[x];
        Food[0] = new FoodItem(" ", $.$$);
        .
        .
        .
        .
        .
        // each '.' represents a space for a possible food element in the array
        addFoods (adtBag, Food);
    }
    private static void addFoods (BagInterface<FoodItem> adtBag, FoodItem[] content) {
    /*as long as the bag is not full, this method will add the food items designated above to
    the bag it will then call the displayBag method to display the contents of the bag.*/
    }
    private static void displayBag(BagInterface<FoodItem> adtBag) {
       //this method will display all the food items in the bag to the user's screen.
    }
}
```