Assignment 4 Writeup

The goal of this program is to find a traversal for each dream candy list given as csv file. The user input (csv file names) is accepted as command-line arguments to the main method of HalloweenNeighborhoodTraversal class. The HalloweenNeighborhoodTraversal class calls the main controller to start the program.

Both visitor pattern and interpreter pattern are implemented. The difference between the two reflects on the implementation of households and candies.

The classes that are shared between the two patterns are summarized here:

CmdHandler.class

- Validates the format of command-line arguments
- Returns the csv file names

IOLibrary.class

- Reads a csv file and converts the content to a string
- Writes a string to an output directory

CsvParser.class

- Parses the csv file to a string array of candy names (for example, {“super size twix”, “snickers”, “fun size mars”})
- Returns the string array

NeighborhoodTraversal.class

- Accepts two parameters to the constructor —
  Household[] neighborhood and String[] candyNames
- The neighborhood contains all households that offer candies
- The candyNames are the names of dream candies in ONE csv files. (for example, {“super size twix”, “snickers”, “fun size mars”})
- For each candy name, traverses through the neighborhood until finds the candy. If the candy is found, records the household where it was found and continues on next candy name. If a candy cannot be found, stop the process.
- Returns the traversal if there exists one.

MainController.class

- Validates command-line arguments and parses csv files using CmdHandler and CsvParser.
- For each csv file, constructs a new NeighborhoodTraversal object with a neighborhood and the candy names. If there exists a traversal, saves it to an output file; if there is no such traversal, prints to console a message. Then continues on next csv file.
- *Notes: the neighborhood can be implemented using either visitor pattern or interpreter pattern.

Besides the classes above, the sizes of candy are also shared between two design patterns. There are four concrete Size classes, SuperSize, KingSize, RegularSize and FunSize.

Design patterns:

There are four types of households that implemented the Household interface. The Household interface provides a method hasCandy(String candyName) that returns true if this household offers the candy, and false otherwise. The implementation of hasCandy(String candyName) differs between design patterns.

In Visitor pattern, the household accepts a visitor to visit all candies until the visitor finds a dream candy:

```
@Override
public boolean hasCandy(String candyName) {
  return accept(new CandyVisitor(candyName));
}

private boolean accept(GenericVisitor<Boolean> visitor) {
  for (Candy candy : CANDIES) {
    if (candy.accept(visitor)) {
      return true;
    }
  }
  return false;
}
```

In Interpreter pattern, the household checks all candies until it finds the dream candy:

```
@Override
public boolean hasCandy(String candyName) {
  for (Candy2 candy : CANDIES) {
    if (candy.typeAndSizeCheck(candyName)) {
      return true;
    }
  }
  return false;
}
```

The implementations of candy classes are also different depending on which design pattern is used. There are ten concrete candy class, each one represents one type of candy.

In Visitor pattern, the Candy class has an accept method to accept the visitor. The visit(Candy candy) method is implemented in the CandyVisitor class. There is no need to implement visit method for all concrete candy classes, because the job it does is the same despite the candy type.

In Interpreter pattern, the typeAndSizeCheck() method is implemented in each of the concrete candy classes. The household classes and candy classes that end with “2” are classes implemented using interpreter pattern.

Please see UML diagrams in the folder of assignment 4.