Coding Assignment 5

Fall 2021

You are creating a multi threaded Trick Or Treating program.

Step 1 – Input files

You will need to create 3 input files for your program. I have attached samples of all 3 files to the assignment. You should create your own files for testing. A different set of files will be for testing during the grading process.

A file of trick or treater's names - I suggest you use short names (10 characters per name) to make the display work well. Sample file is named **`TOT.txt`**.

A file of houses – this is just a list of names that will be displayed as the houses visited by the trick or treaters. Sample file is named **`HOUSES.txt`**. Names should be under 10 characters and the file should only contain at most 6 names.

A file of candy where each line is numbered. Each line contains a candy name followed by a pipe symbol (|) and then its number. There can be as many candies listed in the file as you want – just make sure you have included all numbers from 1 to the number of lines in your file. Sample file is named **`CANDY.txt`**.

Step 2 – Command line arguments

Using the code you created earlier, get the filenames from the command line. For grading consistency, you must set the Candy file as the first file in the command line, the Houses file as the second file and the TrickOrTreaters file as the 3rd file.

CANDY=CANDY.txt HOUSE=HOUSES.txt TOT=TOT.txt

As before, you should **NOT** hardcode the location of the = in the command line argument. Check if all 3 filenames are present in the command line. If any are missing, print the message seen in the sample and exit.

Step 3 – Create Candy List

Create a `HashMap` to hold the candy list. The number from the file should be the map's value and the candy's name should be the key for the map. Open the candy file and extract the information from each line to create a key,value pair that you will then put into the `HashMap`. You will need to use `split()` to parse each line.

```
/* no return */ CreateCandyList(/*pass in the Candy filename and
                                    the Candy List HashMap*/)
{
    /* Set up File and Scanner and declare String array */
    /* Open file and catch exception if needed.  Exit if file does not open */

    /* while reading the file
    /*    use split() to put file line into String array */
    /*    Use map method put() to add (candyname, candynumber) to HashMap */
    /* close the file */
}
```

Step 4 – Create House List

Read house file and create an `ArrayList` of Houses (CandyHouse and ToothbrushHouse).

```
/* return String */ CreateHouses(/* pass in House filename and
                                    ArrayList of type House and
                                    the Candy List HashMap */)
{
    /* Create a string named HouseHeading and set it equal to "           "  */

    /* Set up File and Scanner */
    /* Open file and catch exception if needed.  Exit if file does not open */

    /* while reading the file */
    /*    get next line */
    /*    concatenate HouseHeading with line from file */
          for (int i = 0; i < 11-FileLine.length(); i++)
          {
              HouseHeading += " ";
          }
    /*    if a random number between 0 and 2 is 0 */
    /*    then add a CandyHouse to the ArrayList of Houses */
    /*    else add a ToothbrushHouse to the ArrayList of Houses */

    /* close the file */
    /* concatenate HouseHeading and 2 newlines */
    /* return HouseHeading */
}
```

Step 5 – Create TrickOrTreater List

Read trick or treater file and create an `ArrayList` of TrickOrTreaters.

```
/* no return */ CreateTOTs(/* pass in TOT filename,
                            ArrayList of type TrickOrTreater,
                            ArrayList of Houses)
{
    /* Set up File and Scanner */
    /* Open file and catch exception if needed.  Exit if file does not open */

    /* while reading the file */
    /*    get next line */
    /*    add TrickOrTreater to ArrayList */

    /* close file */
}
```

Step 6 – main()

Get command line arguments

Create Candy list

Create House list

Create TrickOrTreater list

```
ExecutorService executorService = Executors.newCachedThreadPool();
```

enhanced for loop over TrickOrTreater `ArrayList`

```
{
    executorService.execute(it);   // where it is iterator over the ArrayList
}
```

```
TextAreaFrame TAF = new TextAreaFrame();
TAF.setVisible(true);
```

while (TrickOrTreater's `ImDone` static variable is not equal to the size of TrickOrTreater `ArrayList`**)**

{

```
    ScreenOutput = String.format("%s", HouseHeading);
```

enhanced for loop over TrickOrTreater `ArrayList`

```
    {
        ScreenOutput += String.format("%s%s\n", it.getPath(), it.getName());
    }
    TAF.textArea.setText(ScreenOutput);
```

}

```
executorService.shutdown();
```

Create a string called `BucketContents` and put "Candy!!" and two newlines in it.

enhanced for loop over TrickOrTreater ArrayList

concatenate `BucketContents` and the return value of `printBucket()`

```
TAF.textArea.setText(BucketContents);
```

Step 7 – Creating `House`

Create a class named `House`. This class needs to be abstract. It will have a `private String` instance variable named `houseName` and a `HashMap` of type `String`, `Integer` named `CandyList`. The constructor will set `houseName` to the passed in name and `CandyList` to the passed in `CandyList`. This class will also contain an abstract method named `ringDoorbell` that takes a parameter of a `TrickOrTreater`.

Step 8 – Creating `CandyHouses` and `ToothbrushHouses`

Create a class named `CandyHouse` and a class named `ToothbrushHouse`. Both inherit from `House`. Both will override `House`'s `ringDoorbell`.

```
public synchronized String ringDoorbell(TrickOrTreater TOT)
{
    /* Create a string named Candy */
    /* Call addToPath from TrickOrTreater and pass "+"  */

    /* Call Thread.sleep to sleep for 3000 milliseconds */
    /* get random number between 1 and number of candies listed in the CandyList */
```

```
    /* use an enhanced for loop to loop over the CandyList – Week12 – Slides 72-76 */
    {
        /* if the iterator's value equals the chosen random number */
        /* then set Candy equal to key from the iterator */
    }

    /* return Candy */
}
```

A `ToothbrushHouse` should add a dash (instead of +) to the path. It should sleep the same amount. It should NOT pick a candy – it should just return "`Toothbrush`".

Step 9 – Create `TextAreaFrame.java`

Add a new class to your project and name it **exactly** `TextAreaFrame`. After the package statement, add this code **exactly**

```
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class TextAreaFrame extends JFrame
{
    public JTextArea textArea;

    public TextAreaFrame()
    {
        super("TrickOrTreat");
        textArea =  new JTextArea(50, 10);
        textArea.setEditable(false);
        textArea.setFont(new Font("monospaced", Font.PLAIN, 20));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(1100,600);
        setLocationRelativeTo(null);

        setLayout(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();

        constraints.insets = new Insets(10, 10, 10, 10);
        constraints.anchor = GridBagConstraints.WEST;
        constraints.gridx = 0;
        constraints.gridy = 1;
        constraints.gridwidth = 2;
        constraints.fill = GridBagConstraints.BOTH;
        constraints.weightx = 1.0;
        constraints.weighty = 1.0;

        add(new JScrollPane(textArea), constraints);
    }
}
```

Step 10 – Creating `TrickOrTreater`

Create the class TrickOrTreater.

```
public class TrickOrTreater implements Runnable
{
    public static int ImDone;

    private String name;
    private String path = ".";
    private ArrayList<House>ListOfHouses = new ArrayList<>();
    private HashMap<String, Integer>Bucket = new HashMap<>();

    /* constructor accepts a name and an ArrayList of Houses */

    /* instance method getName() that returns the name */

    /* instance method getPath() that return the path */

    /* void instance method addToPath that accepts a string that it concatenates */
    /* onto instance variable path                                               */

    /* private void instance method Walk that accepts a speed */
        /* for loop 10 times                                  */
        /*    concatenate . onto path                         */
        /*    sleep for speed number of milliseconds          */

    /* public instance method named printBucket that returns a String */
    {
        /* declare two strings – Candy and BucketContents                */
        /* declare an integer named CandyCount and set to 0              */

        BucketContents = String.format("%-10s - ", name);

        /*  use an enhanced for loop to loop over Bucket – Week12 – Slides 72-76 */
        {
            /* set Candy equal to the iterator's key */
            /* set CandyCount equal to the iterator's value */
            BucketContents += String.format("%15s %d ", Candy, CandyCount);
        }
        BucketContents += "\n";
        return BucketContents;
    }

    /* override run() from Runnable *.
    {
        /* create integer speed and count and set both to 0 */
        /* create string named Candy */

        /* enhanced for loop over ListOfHouses
        {
            /* set speed equal to a random number between 1 and 1501 */
            /* pass speed to instance method Walk */
            /* call ringDoorbell using the iterator and pass in this */
```

```
            /* store the return value of ringDoorbell() in Candy */

            /* if HashMap Bucket contains a key of Candy */
               /* set count equal to the value associated with Candy from Bucket */
               /* increment count and put (Candy, count) pair in Bucket */
            /* else Candy is not in Bucket - put Candy in Bucket with a value of 1 */
         }

         /* add a synchronized block on object this to protect the update of ImDone */
      }

}
```