Chase Brown

CSCI 3415-001

Program 2 Java

The purpose of this program was to learn some of the basics of Java with a focus around reading in csv files. Using three different csv files from the Marvel Universe the goal is to read the three files and parse them to find out some basic data. There were five goals that we were looking to achieve. Finding the number of comics, the number of heroes, the mean of books per character, the mean characters per book, and the mean partners per hero.

```
"C:\Program Files\Java\jdk-9\bin\java" "-javaagent:C:\Program Files
Number of Comics: 12651
Number of Heros: 6439
Mean books per Character: 14.93
Mean Characters per book: 1.96
The mean number of partners per hero: 89.22
The mean number of partners per hero without duplicates: 34.82

Process finished with exit code 0
```

I implemented part 5 the number of partners in two different ways. I first print out the mean number of partners counting duplicates (hero1/hero2 can happen more than once). I then also printed off a non-duplicate number meaning that it only counts the first time that hero1/hero2 happens. I did this by overriding the hashCode() function of java for my partners class and the equals(). I then created a HashSet(). This prevents any duplicates because with a HashSet() if the hash is the same it will not insert the data into the Set like a List will. Most of the implementation of the code was simple. Use of super-csv makes parsing the csv files very simple with a few lines of code. I parsed the node and type file to find the number of comics and heroes. I then used these numbers when parsing the other files to find the means of the required data.

I have been programming in Java a lot lately due to a project that I am working on so this program was very simple for me to code. I decided to use Super-csv for this program because I usually use open-csv and wanted to test out the difference in them. I did find that Super-csv is easier to code than open with minimal differences for a simple parse like this program required. Super-csv does have some nice documentation(http://super-csv.github.io/super-csv/index.html) and was the only reference that I used for this project. I like coding with Java due to experience with C++ and the fact that they are very close in code style. While I didn't learn a lot about Java that I don't already know with this project it gave me an opportunity to test out Super-csv.

```java
import java.io.FileReader;
import java.util.*;
import org.supercsv.cellprocessor.constraint.NotNull;
import org.supercsv.cellprocessor.ift.CellProcessor;
import org.supercsv.io.CsvBeanReader;
import org.supercsv.io.ICsvBeanReader;
import org.supercsv.prefs.CsvPreference;

class Main{
    private static CellProcessor[] getProcessors(){
        return new CellProcessor[]{
                new NotNull(), //column 1
                new NotNull(), // column 2
        };
    }
    //check the list to see if it is type comic or type hero.
    private static int containsComics(List<Comic> list){
        int comicCount = 0;
        for(Comic e : list){
            String comicString = e.getType();
            if(Objects.equals(comicString, "comic")) {
                comicCount++;
            }
        }
        return comicCount;
    }

    private static void numberofcomics() throws Exception{
        ICsvBeanReader beanReader = null;
        try{
            beanReader = new CsvBeanReader(new FileReader("./resources/nodes.csv"),
                    CsvPreference.STANDARD_PREFERENCE);
            final String[] header = beanReader.getHeader(true);
            final CellProcessor[] processors = getProcessors();

            List<Comic> comics = new ArrayList<>();
            Comic c;
            while((c = beanReader.read(Comic.class, header, processors)) != null){
                comics.add(c);
            }
            int comicCount = containsComics(comics);
            int heroCount = comics.size() - comicCount;
            System.out.println(String.format("Number of Comics: %d  \nNumber of Heros:
%d", comicCount, heroCount));
            avgbooks(heroCount);
            float CPB = (float)comicCount/heroCount;
            System.out.println(String.format("Mean Characters per book: %.2f", CPB));
            avgPartners(heroCount);
            avgPartnersnodup(heroCount);

        }finally{
            if(beanReader != null){
                beanReader.close();
            }
        }
    }
    //find the average number of partners..
    private static void avgPartners(int heroCount)throws Exception{
        ICsvBeanReader beanReader = null;
```

```
        try{
            beanReader = new CsvBeanReader(new FileReader("./resources/hero-
network.csv"),
                    CsvPreference.STANDARD_PREFERENCE);
            final String[] header = beanReader.getHeader(true);
            final CellProcessor[] processors = getProcessors();

            List<Partners> partners = new ArrayList<>();
            Partners p;
            while((p= beanReader.read(Partners.class, header, processors)) != null) {
                partners.add(p);
            }
            float partnerswithdup = (float)partners.size()/heroCount;
            System.out.println(String.format("The mean number of partners per hero:
%.2f", partnerswithdup));

        }finally{
            if(beanReader != null){
                beanReader.close();
            }
        }
    }
    //find the average number of partners without duplicates.  hero1/hero2 only
counted once not every time
    private static void avgPartnersnodup(int heroCount)throws Exception{
        ICsvBeanReader beanReader = null;
        try{
            beanReader = new CsvBeanReader(new FileReader("./resources/hero-
network.csv"),
                    CsvPreference.STANDARD_PREFERENCE);
            final String[] header = beanReader.getHeader(true);
            final CellProcessor[] processors = getProcessors();

            Set<Partners> partners = new HashSet<>();
            Partners p;
            while((p = beanReader.read(Partners.class, header, processors)) != null){
                partners.add(p);
            }
            float partnersnodup = (float)partners.size()/heroCount;
            System.out.println(String.format("The mean number of partners per hero
without duplicates: %.2f", partnersnodup));

        }finally{
            if(beanReader != null){
                beanReader.close();
            }
        }
    }
    //Find the average books per hero
    private static void avgbooks(int heroCount)throws Exception{
        ICsvBeanReader beanReader = null;
        try{
            beanReader = new CsvBeanReader(new FileReader("./resources/edges.csv"),
                    CsvPreference.STANDARD_PREFERENCE);
            final String[] header = beanReader.getHeader(true);
            final CellProcessor[] processors = getProcessors();

            List<Hero> books = new ArrayList<>();
            Hero h;
            while((h= beanReader.read(Hero.class, header, processors)) != null){
                books.add(h);
            }
            float count = (float)books.size()/heroCount;
```

```
            System.out.println(String.format("Mean books per Character: %.2f",
count));
        }finally{
            if(beanReader != null){
                beanReader.close();
            }
        }
    }

    public static void main(String args[]) throws Exception {
        //try to read the file else throw exception
        try {
            numberofcomics();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.util.Objects;

public class Partners {
    private String hero1;
    private String hero2;

    public Partners(){}
    public Partners(final String hero, final String partner){
        this.hero1 = hero;
        this.hero2 = partner;
    }

    public String getHero1(){return hero1;}
    public void setHero1(String h){
        this.hero1 = h;
    }

    public String getHero2(){return hero2;}
    public void setHero2(String h) {
        this.hero2 = h;
    }
    //overload the equals
    public boolean equals(Object me){
        Partners PartnersMe = (Partners)me;
        if(Objects.equals(this.hero1, PartnersMe.hero1) && Objects.equals(this.hero2,
PartnersMe.hero2))
            return true;
        else
            return false;
    }
    //override the hashcode function for the class to set the hash number for Set
table.  Prevent counting Dups
    public int hashCode(){
        return Objects.hashCode(this.getHero1() + this.getHero2());
    }
}
```

```java
public class Comic {
    private String node;
    private String type;

    public Comic(){}
    public Comic(final String node, final String type){
        this.node = node;
        this.type = type;
    }

    public String getNode(){return node;}
    public void setNode(String n){this.node = n;}
    public String getType(){return type;}
    public void setType(String s){this.type = s;}
}
```

```java
public class Hero {
    private String hero;
    private String comic;

    public Hero(){}
    public Hero(final String hero, final String comic){
        this.hero = hero;
        this.comic = comic;
    }

    public String getHero(){return hero;}
    public void setHero(String h){
        this.hero = h;
    }
    public String getComic(){return comic;}
    public void setComic(String c){
        this.comic = c;
    }
}
```