CS 5800 - HW1

Ada (Donling) Yang

Github Directory: https://github.com/adayang0046/5800-1.git

Program 1 - Inheritance

Employee.java - Employee and its inheritance classes

```
public class Employee {
    String firstName, lastName, ssn;

    Employee(String firstName, String lastName, String ssn) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.ssn = ssn;

    }

    public void printInfo() {
        System.out.print(firstName + " " + lastName + " | SSN: " + ssn +" ");
    }

}

class SalariedEmployee extends Employee {
    int weeklySalary;

    SalariedEmployee(String firstName, String lastName, String ssn, int weeklySalary) {
        super(firstName, lastName, ssn);
        this.weeklySalary = weeklySalary;
    }

    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("| Weekly Salary: $" + weeklySalary);
    }
}

 class HourlyEmployee extends Employee {
    int wage, hoursWorked;

    HourlyEmployee(String firstName, String lastName, String ssn, int wage, int hoursWorked) {
        super(firstName, lastName, ssn);
        this.wage = wage;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("| Wgae: $" + wage + " | Hours worked: " + hoursWorked);
    }
}

class CommisionEmployee extends Employee {
    int commisionRate, grossSales;

    CommisionEmployee(String firstName, String lastName, String ssn, int commisionRate, int grossSales) {
        super(firstName, lastName, ssn);
        this.commisionRate = commisionRate;
        this.grossSales = grossSales;
    }

    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("| Commision rate: " + commisionRate + "% " + " | Sales: " + grossSales);
    }
}

class BaseEmployee extends Employee {
    int baseSalary;

    BaseEmployee(String firstName, String lastName, String ssn, int baseSalary) {
        super(firstName, lastName, ssn);
        this.baseSalary = baseSalary;
    }

    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("| Base salary: " + baseSalary);
    }
}
~
```

Main.java - Creates all different employees and prints them out.

```java
public class Main {
    // Part 1: Read and store all the input data
    public static void main(String[] args) {
        SalariedEmployee se1 = new SalariedEmployee("Joe","Jones","111-11-1111",2500);

        SalariedEmployee se2 = new SalariedEmployee("Renwa","Chanel","555-55-5555",1700);

        HourlyEmployee he1 = new HourlyEmployee("Stephanie","Smith","222-22-2222",25,32);

        HourlyEmployee he2 = new HourlyEmployee("Mary","Quinn","333-33-3333",19,47);

        CommisionEmployee ce1 = new CommisionEmployee("Nicole","Dior","444-44-4444",15,50000);

        CommisionEmployee ce2 = new CommisionEmployee("Mahnaz","Vaziri","777-77-7777",22,40000);

        BaseEmployee be = new BaseEmployee("Mike","Davenport","666-66-6666", 95000);

        se1.printInfo();
        he1.printInfo();
        he2.printInfo();
        ce1.printInfo();
        se2.printInfo();
        be.printInfo();
        ce2.printInfo();


    }
```

Output - all the employees' information

```
PS D:\5300-01\5300-1\1-IS> java Main
Joe Jones | SSN: 111-11-1111 | Weekly Salary: $2500
Stephanie Smith | SSN: 222-22-2222 | Wgae: $25 | Hours worked: 32
Mary Quinn | SSN: 333-33-3333 | Wgae: $19 | Hours worked: 47
Nicole Dior | SSN: 444-44-4444 | Commision rate: 15%  | Sales: 50000
Renwa Chanel | SSN: 555-55-5555 | Weekly Salary: $1700
Mike Davenport | SSN: 666-66-6666 | Base salary: 95000
Mahnaz Vaziri | SSN: 777-77-7777 | Commision rate: 22%  | Sales: 40000
```

Program 2 - Polymorphism

Ship.java - Constructors and other functions of Ship, Cruise Ship, and Cargo Ship

```
//Ship.java

 class Ship {
    String shipName, yearBuilt;

    public Ship(String shipName, String yearBuilt){
        this.shipName = shipName;
        this.yearBuilt = yearBuilt;
    }

    public String getName() {return shipName;}
    public String getYear() {return yearBuilt;}

    public void setName(String shipName){
        this.shipName = shipName;
    }

    public void setYear(String shipYear){
        this.yearBuilt = shipYear;
    }

    public void printInfo(){
        System.out.println("Ship: " + shipName + ", Built in: " + yearBuilt);

    }

}

class CruiseShip extends Ship{
    int maxPassengers;

    public CruiseShip (String shipName, String yearBuilt, int maxPassengers){
        super(shipName, yearBuilt);
        this.maxPassengers = maxPassengers;

    }

    public int getMax() {return maxPassengers;}
    public void setMax (int maxPassengers){
        this.maxPassengers = maxPassengers;
    }

    @Override
    public void printInfo(){
        System.out.println("Cruise Ship: "+ getName()+", Max Pessagers: " + maxPassengers);

    }
}

class CargoShip extends Ship{
    int capacity;

    public CargoShip (String shipName, String yearBuilt, int capacity){
        super(shipName, yearBuilt);
        this.capacity = capacity;

    }

    public int getCpacity() {return capacity;}
    public void setCapacity(int capacity){
        this.capacity = capacity;
    }

    @Override
    public void printInfo(){
        System.out.println("Cargo Ship: "+ getName()+", Cargo Capacity (in tons): " + capacity);

    }
}
```

Main.java - Create a list that contains 3 different kinds of ships and prints all the ships

```java
//Main.java

public class Main {
    public static void main(String[] args){
        Ship[] shipList = new Ship [3];
        shipList[0]= new Ship("Unnamed little ship","1991");
        shipList[1]= new CruiseShip("Small cruise ship","1845",200);
        shipList[2] =  new CargoShip ("Huge cargo ship","2000",3000000);

        for(Ship i: shipList){
            i.printInfo();
        }


    }
}
```

Output - all the ship info

```
PS D:\5300-01\5300-1\2-POLY> java Main
Ship: Unnamed little ship, Built in: 1991
Cruise Ship: Small cruise ship, Max Pessagers: 200
Cargo Ship: Huge cargo ship, Cargo Capacity (in tons): 3000000
```

Program 3 - Aggregation

Instructor.java - Instructor class

```
public class Instructor {
    private String fristName, lastName, officeNum;

    public void setFirst(String firstName){
        this.fristName = firstName;
    }

    public void setLast(String lastName){
        this.lastName = lastName;
    }

    public void setOffice(String officeNum){
        this.officeNum = officeNum;
    }

    public Instructor(String firstName, String lastName, String officeNum){
        setFirst(firstName);
        setLast(lastName);
        setOffice(officeNum);
    }

    public String getFirst(){
        return fristName;
    }

    public String getLast(){
        return lastName;
    }

    public String getOffice(){
        return officeNum;
    }



    public void print(){

        System.out.println("Instructor: " + fristName + " " + lastName + " | Office number: " + officeNum);
    }

}
~
```

Textbook.java - Textbook class

```
public class Textbook {
    private String title,author,publisher;
    public void setTitle(String title){
        this.title = title;
    }
    public void setAuthor(String author){
        this.author = author;
    }
    public void setPublisher(String publisher){
        this.publisher = publisher;
    }
    public Textbook(String title, String author, String publisher){
        setTitle(title);
        setAuthor(author);
        setPublisher(publisher);
    }

    public String getTitle(){
        return title;
    }

    public String getAuthor(){
        return author;
    }

    public String getPublisher(){
        return publisher;
    }

    public void print(){
        System.out.println("Textbook: " + title + " | Author: " + author + " Publisher: " + publisher);
    }


}
~
```

Course.java - Course class

```
import javax.swing.text.html.HTMLDocument.Iterator;

public class Course {
    private String course;
    private Instructor[] instructor;
    private Textbook[] textbook;

    public Course(String course, Instructor[] instructor, Textbook[] textbook){
        this.course = course;
        this.instructor = instructor;
        this.textbook = textbook;
    }

    public void print() {
        System.out.println("\nCourse name: " + course);
        for(Instructor i : instructor){
            i.print();
        }
        for( Textbook b: textbook){
        b.print();
        }
    }


}
~
```

Main.java - Course with 1 instructor and 1 textbook + course with 2 instructor and 2 textbooks

```java
import java.util.concurrent.CountDownLatch;

public class Main {
    public static void main(String[] args){

        //Create all instructors and textbook needed

        Instructor instructor1 = new Instructor("Nima","Davarpanah","3-2636");
        Instructor instructor2 = new Instructor("Luke","Smith","3-2638");
        Textbook textbook1 = new Textbook("Clean Code","Robert C. Martin", "Prentice Hall");
        Textbook textbook2 = new Textbook("Finding Patterns","John D. Lee", "Cal Poly");

        //Course with one instructor and one textbook
        Instructor[] ins1 = {instructor1};
        Textbook[] tex1 = {textbook1};
        Course course1 = new Course("CS-5800",ins1, tex1);
        course1.print();

        Instructor[] ins2 = {instructor1,instructor2};
        Textbook[] tex2 = {textbook1,textbook2};
        Course course = new Course("SE", ins2,tex2);
        course.print();
    }
}
```

Output - the print out of both courses

```
PS D:\5300-01\5300-1\3-HAS> java Main

Course name: CS-5800
Instructor: Nima Davarpanah | Office number: 3-2636
Textbook: Clean Code | Author: Robert C. Martin Publisher: Prentice Hall

Course name: SE
Instructor: Nima Davarpanah | Office number: 3-2636
Instructor: Luke Smith | Office number: 3-2638
Textbook: Clean Code | Author: Robert C. Martin Publisher: Prentice Hall
Textbook: Finding Patterns | Author: John D. Lee Publisher: Cal Poly
```

Program 4 - Composition

File.java - The File class

```java
//File.java
//
public class File {

    private String fileName;

    public void setFileName(String fileName){
        this.fileName = fileName;
    }

    public File(String fileName){
        setFileName(fileName);
    }

    public String getFileName(){
        return fileName;
    }

    // to print file based on its level --> int start is file's #layer based on the root folder
    public void print(int start){
        String sep = "| ".repeat(start);
        System.out.println(sep +"File: " + fileName);
    }
}
```

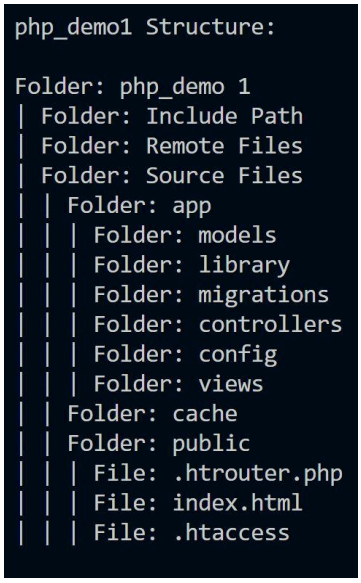
Folder.java - the Folder class

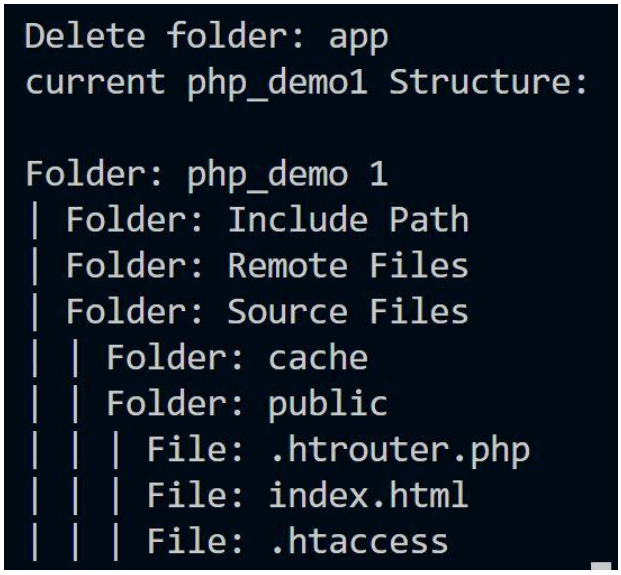
```java
//Folder.java
//Using hash map to store sub folder and file list

import java.util.HashMap;
import java.util.Map;

public class Folder {
    private String folderName;
    private Map<String, File> fileList;
    private Map<String, Folder> folderList;

    public void setFolderName(String folderName){
        this.folderName = folderName;
    }

    //Initiate an empty folder with a name and empty sub folder and file list.
    public Folder(String folderName){
        setFolderName(folderName);
        this.fileList = new HashMap<>();
        this.folderList = new HashMap<>();

    }

    public String getFolderName(){
        return folderName;
    }

    public Folder getSubFolder(String subFolderName){
        return folderList.get(subFolderName);
    }

    public void addFile(File file) {
        fileList.put(file.getFileName(),file);
    }

    public void addFolder(Folder folder){
        folderList.put(folder.getFolderName(),folder);
    }

    //delete file and folder from the current source folder
    public File deleteFile(String fileName){
        return fileList.remove(fileName);
    }

    public Folder deleteFolder(String folderName){
        return folderList.remove(folderName);
    }

    //add a list of folders or files
    public void addFolderList (String[] folderNames){
        for (String i : folderNames){
            Folder temp = new Folder(i);
            this.addFolder(temp);
        }
    }

    public void addFileList (String[] fileNames){
        for (String i : fileNames){
            File temp = new File(i);
            this.addFile(temp);
        }
    }
    // print the current folder structure using the current folder as the source
    public void print(int start){
        String sep = "| ".repeat(start);
        System.out.println(sep + "Folder: " + folderName);
        for(File i : fileList.values()){
            i.print(start +1);
        }

        for(Folder i : folderList.values()){

            i.print(start +1);

        }
    }


}
~
```

Main.java - Creates all the folders and files, links them into the preferred structure, and prints them (all first, app deleted second)

```
-
public class Main {
    public static void main(String[] args){
        //adding all folders and files and linking all of them in to a tree
        Folder phpDemo1 = new Folder("php_demo 1");

        String[] phpSubFolders = {"Source Files"};
        String[] sourceSubFolders = {"app","cache","public"};
        phpDemo1.addFolderList(phpSubFolders);
        Folder source  = phpDemo1.getSubFolder("Source Files");
        source.addFolderList(sourceSubFolders);
        Folder app = source.getSubFolder("app");
        Folder pub = source.getSubFolder("public");
        String[] appSubFolders = {"config","controllers","library","migrations","models","views"};
        String[] publicSubFiles = {".htaccess",".htrouter.php","index.html"};
        app.addFolderList(appSubFolders);
        pub.addFileList(publicSubFiles);
        String[] phpSubFolders2 = {"Include Path","Remote Files"};
        phpDemo1.addFolderList(phpSubFolders2);

        //print the structure
        System.out.println("\nphp_demo1 Structure: \n");
        phpDemo1.print(0);

        //delete the app folder and print the structure again
        source.deleteFolder("app");
        System.out.println("\nDelete folder: app\ncurrent php_demo1 Structure: \n");
        phpDemo1.print(0);
    }
}
~
~
```

Output - 1: the full tree, 2: the tree after app is deleted


php_demo1 Structure:
Folder: php_demo 1
| Folder: Include Path
| Folder: Remote Files
| Folder: Source Files
| | Folder: app
| | | Folder: models
| | | Folder: library
| | | Folder: migrations
| | | Folder: controllers
| | | Folder: config
| | | Folder: views
| | Folder: cache
| | Folder: public
| | | File: .htrouter.php
| | | File: index.html
| | | File: .htaccess


Delete folder: app
current php_demo1 Structure:
Folder: php_demo 1
| Folder: Include Path
| Folder: Remote Files
| Folder: Source Files
| | Folder: cache
| | Folder: public
| | | File: .htrouter.php
| | | File: index.html
| | | File: .htaccess

END.