Daniel Plascencia

Repository Link: <https://github.com/ExodiusLux/CS-4800/tree/main/Homework_1>

**Homework 1**

1. **INHERITANCE – “IS A” Relationship**

Create the classes in the inheritance hierarchy given below. Each class should have appropriate constructors, set methods and get methods, also known as accessors and mutators respectively.

An **Employee** **class** should have a first name, last name and social security number.

A **SalariedEmployee** **class** should have a weekly salary.

A **HourlyEmployee** **class** should have a wage and a number of hours worked.

A **CommisionEmployee** **class** should have a commission rate and gross sales.

A **BaseEmployee** **class** should have a base salary.

Finally, write a driver program, that is a program that will instantiate each of the classes above and populate with the following information:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First name | Last name | Social Sec # | Weekly  Salary | Wage | Hours worked | Com rate | Gross salary | Base salary |
| Joe | Jones | 111-11-1111 | $2,500 |  |  |  |  |  |
| Stephanie | Smith | 222-22-2222 |  | $25 | 32 |  |  |  |
| Mary | Quinn | 333-33-3333 |  | $19 | 47 |  |  |  |
| Nicole | Dior | 444-44-4444 |  |  |  | 15% | $50,000 |  |
| Renwa | Chanel | 555-55-5555 | $1,700 |  |  |  |  |  |
| Mike | Davenport | 666-66-6666 |  |  |  |  |  | $95,000 |
| Mahnaz | Vaziri | 777-77-7777 |  |  |  | 22% | $40,000 |  |

public class BaseEmployee extends Employee{

    int baseSalary;

    BaseEmployee(String firstname, String lastname, String SSN, int baseSalary){

        super(firstname, lastname, SSN);

        this.baseSalary = baseSalary;

    }

    BaseEmployee(){

        super("", "", "");

        baseSalary = 0;

    }

    public void print(){

        System.out.println(first\_name + " " + last\_name + " SSN:" + SSN + " Salary:" + baseSalary);

    }

}

public class ComissionEmployee extends Employee{

    int comissionRate;

    int grossSales;

    ComissionEmployee(String firstname, String lastname, String SSN, int comissionRate, int grossSales){

        super(firstname, lastname, SSN);

        this.comissionRate = comissionRate;

        this.grossSales = grossSales;

    }

    ComissionEmployee(){

        super("", "", "");

        comissionRate = 0;

        grossSales = 0;

    }

    public void print(){

        System.out.println(first\_name + " " + last\_name + " SSN:" + SSN + " Comission Rate:" + comissionRate + " Gross Sales:" + grossSales);

    }

}

public class Employee {

    String first\_name;

    String last\_name;

    String SSN;

    Employee(String first\_name, String last\_name, String SSN){

        this.first\_name = first\_name;

        this.last\_name = last\_name;

        this.SSN = SSN;

    }

    Employee(){

        first\_name = "";

        last\_name = "";

        SSN = "";

    }

}

public class HourlyEmployee extends Employee{

    int wage;

    int hoursWorked;

    HourlyEmployee(String firstname, String lastname, String SSN, int wage, int hoursWorked){

        super(firstname, lastname, SSN);

        this.wage = wage;

        this.hoursWorked = hoursWorked;

    }

    HourlyEmployee(){

        super("", "", "");

        wage = 0;

        hoursWorked = 0;

    }

    public void print(){

        System.out.println(first\_name + " " + last\_name + " SSN:" + SSN + " Wage:" + wage + " Hours Worked:" + hoursWorked);

    }

}

public class SalariedEmployee extends Employee{

    int weekly\_salary;

    SalariedEmployee(String firstname, String lastname, String SSN, int weekly\_salary){

        super(firstname, lastname, SSN);

        this.weekly\_salary = weekly\_salary;

    }

    SalariedEmployee(){

        super("", "", "");

        weekly\_salary = 0;

    }

    public void print(){

        System.out.println(first\_name + " " +last\_name + " SSN:" + SSN + " Weekly\_Salary:" +weekly\_salary);

    }

}

public class Inherit\_driver {

    public static void main(String[] args){

        SalariedEmployee Joe\_Jones = new SalariedEmployee("Joe", "Jones", "111-11-1111", 2500);

        HourlyEmployee Stephine\_Smith = new HourlyEmployee("Stephine", "Smith", "222-22-2222", 25,32 );

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

        ComissionEmployee Niccole\_Dior = new ComissionEmployee("Niccole", "Dior", "444-44-4444", 15, 50000);

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

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

        ComissionEmployee Mahnaz\_Vaziri = new ComissionEmployee("Mahnaz", "Vaziri", "777-77-7777", 22, 40000);

        Joe\_Jones.print();

        Stephine\_Smith.print();

        Mary\_Quinn.print();

        Niccole\_Dior.print();

        Renwa\_Chanel.print();

        Mike\_Davenport.print();

        Mahnaz\_Vaziri.print();

    }

}

Output:

A computer screen with blue text

Description automatically generated

1. **POLYMORPHISM**

Design a **Ship class** that has the following members:

1. A member variable for the name of the ship(String)
2. A member variable for the year that the ship was built(String)
3. Constructor with all setters and getters(Also known as accessors and mutators)
4. A **print function** that will display the ship name and year it was built.

Design a **CruiseShip class** that is derived from the **Ship class** and has the following members:

1. A member variable for the maximum number of passengers(Int)
2. Constructor with all setters and getters(Also known as accessors and mutators)
3. A **print function** that overrides the base print function and will display the ship name and maximum number of passengers.

Design a **CargoShip class** that is derived from the **Ship class** and has the following members:

1. A member variable for the cargo capacity in tonnage(Int)
2. Constructor with all setters and getters(Also known as accessors and mutators)
3. A **print function** that overrides the base print function and will display the ship name and ships cargo capacity.

Demonstrate the classes in a program that has a static **array of Ship** size of 3.. The array elements should be initiated dynamically with one type each of **Ship**, **CruiseShip**, and **CargoShip objects**. Create a loop that iterates through the array calling each print function.

public class CargoShip extends Ship{

    int cargo\_capacity;

    CargoShip(String ship\_name, String year\_built, int cargo\_capacity){

        super(ship\_name, year\_built);

        this.cargo\_capacity = cargo\_capacity;

    }

    public int getCargoCapacity() {

        return cargo\_capacity;

    }

    public void setCargoCapacity(int cargo\_capacity) {

        this.cargo\_capacity = cargo\_capacity;

    }

    @Override

    public void print() {

        super.print();

        System.out.println("Cargo Capacity: " + cargo\_capacity + " tons");

    }

}

public class CruiseShip extends Ship {

    int max\_passengers;

    CruiseShip(String ship\_name, String year\_built, int max\_passengers){

        super(ship\_name, year\_built);

        this.max\_passengers = max\_passengers;

    }

    public int getMaxPassengers() {

        return max\_passengers;

    }

    public void setMaxPassengers(int max\_passengers) {

        this.max\_passengers = max\_passengers;

    }

    @Override

    public void print() {

        super.print();

        System.out.println("Max Passengers: " + max\_passengers);

    }

}

public class Ship {

    String ship\_name;

    String year\_built;

    Ship(String ship\_name, String year\_built){

        this.ship\_name = ship\_name;

        this.year\_built = year\_built;

    }

    public void print(){

        System.out.println("Ship Name: " + ship\_name );

        System.out.println("Year Built: " + year\_built);

    }

    public String getName() {

        return ship\_name;

    }

    public void setName(String ship\_name) {

        this.ship\_name = ship\_name;

    }

    public String getYearBuilt() {

        return year\_built;

    }

    public void setYearBuilt(String year\_built) {

        this.year\_built = year\_built;

    }

}

public class ship\_driver {

    public static void main(String[] args) {

        Ship[] ship\_arr = new Ship[3];

        Ship base\_ship = new Ship("Admire", "1997");

        CruiseShip cruise\_ship = new CruiseShip("Carnival Cruise", "2019", 3000);

        CargoShip cargo\_ship = new CargoShip("Tankers", "2011", 200000);

        ship\_arr[0] = base\_ship;

        ship\_arr[1] = cruise\_ship;

        ship\_arr[2] = cargo\_ship;

        for(Ship ship: ship\_arr){

            ship.print();

            System.out.println();

        }

    }

}

Output:

A black background with blue lines

Description automatically generated

1. **AGGREGATION -** **“HAS A” Relationship**

You will create a Course class that “has a” Instructor and “has a” textbook. The Instructor and textbook should be there own type(That means each is a class). In the Course class you will instantiate the instructor and textbook objects as needed.

Create a **Course Class** which holds the following information:

1. Course *name*(private String)
2. An instructors first name, last name and office number. This information is accessed using the setter and getters of the instructor class.
3. The textbooks title, author and publisher. This information is accessed using the setter and getters of the texbooks class.
4. Print function that prints out Course name, instructor first and last name and the text book title and author.
5. Create a driver program to show this information. You can use our course information: Instructor Nima Davarpanah office 3-2636, textbook Clean Code…etc
6. Now modify the code to have two instructors and two textbooks.
7. Create a driver program to show this information, make up the other information.

public class Textbook {

    private String title;

    private String author;

    private String publisher;

    Textbook(String title, String author, String publisher){

        this.title = title;

        this.author = author;

        this.publisher = publisher;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getAuthor() {

        return author;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    public String getPublisher() {

        return publisher;

    }

    public void setPublisher(String publisher) {

        this.publisher = publisher;

    }

}

public class Instructor {

    private String first\_name;

    private String last\_name;

    private String office\_number;

    Instructor(String first\_name, String last\_name, String office\_number){

        this.first\_name = first\_name;

        this.last\_name = last\_name;

        this.office\_number = office\_number;

    }

    public String getFirstName() {

        return first\_name;

    }

    public void setFirstName(String first\_name) {

        this.first\_name = first\_name;

    }

    public String getLastName() {

        return last\_name;

    }

    public void setLastName(String last\_name) {

        this.last\_name = last\_name;

    }

    public String getOfficeNumber() {

        return office\_number;

    }

    public void setOfficeNumber(String office\_number) {

        this.office\_number = office\_number;

    }

}

public class Course {

    private String course\_name;

    private Instructor instructor;

    private Textbook textbook;

    Course(String course\_name, Instructor instructor, Textbook textbook){

        this.course\_name = course\_name;

        this.instructor = instructor;

        this.textbook = textbook;

    }

    public void printInfo() {

        System.out.println("Course Name: " + course\_name);

        System.out.println("Instructor: " + instructor.getFirstName() + " " + instructor.getLastName());

        System.out.println("Office Number: " + instructor.getOfficeNumber());

        System.out.println("Textbook: " + textbook.getTitle() + " by " + textbook.getAuthor());

    }

}

public class Course\_driver {

    public static void main(String[] args) {

        Instructor instructor1 = new Instructor("Nima", "Davarpanah", "3-2636");

        Instructor instructor2 = new Instructor("Justin", "Royce", "4-1234");

        Textbook textbook1 = new Textbook("Clean Code", "Robert C. Martin", "Prentice Hall");

        Textbook textbook2 = new Textbook("Design Patterns Elements of Reusable Object-Oriented Software", "Erich Gamma", "Addison Weekly");

        Course course1 = new Course("Software Engineering", instructor1, textbook1);

        Course course2 = new Course("Object Oriented", instructor2, textbook2);

        System.out.println("Course 1 Information:");

        course1.printInfo();

        System.out.println();

        System.out.println("Course 2 Information:");

        course2.printInfo();

    }

}

Output:

A black screen with blue text

Description automatically generated

1. **COMPOSITION - “OWN A” Relationship**

You will create a **Folder** **class** and a **File class** that will represent the File System on your computer. They should have all the proper member variables, setters and getters. The folder class will have a print function that prints out all subfolders and files. The file class will have a print function that prints out the name of the file.

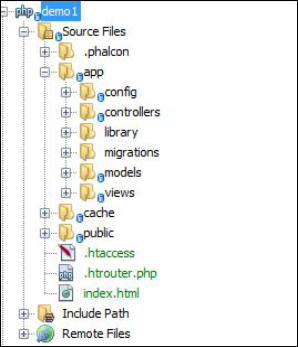
A Folder can contain zero or more Files. A Folder can also contain zero or more Sub-Folders.

A File must be stored within a folder.

If a folder is deleted all the files and sub-folders in the folder are also deleted.

In the structure below we see **php\_demo1** is a folder and has a sub-folder of **Source Files.**

Recreate the structure below:



1. As part of the driver program print out the php\_demo1 folder and all sub folders. All sub folders should also print out their content until the full structure above is printed.
2. Now delete the folder **app** and print out the full structure.
3. Now delete the folder **public** and print out the full structure.

public class File {

    private String name;

    public File(String name) {

        this.name = name;

    }

    public String getName() {

        return name;

    }

    public void print() {

        System.out.println(name);

    }

}

public class Folder {

    private String name;

    private List<Folder> sub\_folders;

    private List<File> files;

    public Folder(String name) {

        this.name = name;

        this.sub\_folders = new ArrayList<>();

        this.files = new ArrayList<>();

    }

    public String getName() {

        return name;

    }

    public List<Folder> getSubFolders() {

        return sub\_folders;

    }

    public List<File> getFiles() {

        return files;

    }

    public void addSubFolder(Folder folder) {

        sub\_folders.add(folder);

    }

    public void addFile(File file) {

        files.add(file);

    }

    public void print() {

        printFolderStructure(this, 0);

    }

    private void printFolderStructure(Folder folder, int depth\_space) {

        StringBuilder indents = new StringBuilder();

        for (int i = 0; i < depth\_space; i++) {

            indents.append("  ");

        }

        System.out.println(indents.toString() + folder.getName());

        for (File file : folder.getFiles()) {

            System.out.println(indents.toString() + "  " + file.getName());

        }

        for (Folder subFolder : folder.getSubFolders()) {

            printFolderStructure(subFolder, depth\_space + 1);

        }

    }

}

public class Folders\_driver {

    public static void main(String[] args) {

        //Parent File

        Folder phpDemo1 = new Folder("php\_demo1");

        //SubFiles of phpDemo1

        Folder SourceFiles = new Folder("Source Files");

        Folder IncludePath = new Folder("Include Path");

        Folder RemoteFiles = new Folder("Remote Files");

        //Subfiles of Source Files

        Folder phalcon = new Folder(".phalcon");

        Folder app = new Folder("app");

        Folder cache = new Folder("cache");

        Folder public\_folder = new Folder ("public");

        //Subfolders of app

        Folder config = new Folder("config");

        Folder controllers = new Folder("controllers");

        Folder library = new Folder("library");

        Folder migrations = new Folder("migrations");

        Folder models = new Folder("models");

        Folder views = new Folder("views");

        //Files of public

        File htaccess = new File(".htaccess");

        File htrouter\_php = new File(".hrouter.php");

        File index\_html = new File("index.html");

        phpDemo1.addSubFolder(SourceFiles);

        phpDemo1.addSubFolder(IncludePath);

        phpDemo1.addSubFolder(RemoteFiles);

        SourceFiles.addSubFolder(phalcon);

        SourceFiles.addSubFolder(app);

        SourceFiles.addSubFolder(cache);

        SourceFiles.addSubFolder(public\_folder);

        app.addSubFolder(config);

        app.addSubFolder(controllers);

        app.addSubFolder(library);

        app.addSubFolder(migrations);

        app.addSubFolder(models);

        app.addSubFolder(views);

        public\_folder.addFile(htaccess);

        public\_folder.addFile(htrouter\_php);

        public\_folder.addFile(index\_html);

        phpDemo1.print();

        System.out.println();

        SourceFiles.getSubFolders().remove(app);

        phpDemo1.print();

        System.out.println();

        SourceFiles.getSubFolders().remove(public\_folder);

        phpDemo1.print();

    }

}

Output:

A computer screen with text on it

Description automatically generated