

**BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM
KHÓA CÔNG NGHỆ ĐIỆN TỬ**



**TIỂU LUẬN KẾT THÚC HỌC PHẦN
HỌC KỲ I, 2021-2022.**

**TÊN HỌC PHẦN:
Lập Trình Hướng Đối Tượng**

Họ và tên sinh viên:	Phạm Minh Tuấn
Mã số sinh viên:	19469421
Tên lớp học phần:	DHDTMT15A
Mã lớp học phần:	420300359101
Email liên hệ:	mtp92621@gmail.com
Số điện thoại:	0931890287
Giảng viên giảng dạy:	Trần Hồng Vinh

Tp HCM, tháng 12 năm 2021

1. TỔNG QUÁT VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

1.1 Khái niệm

Lập trình hướng đối tượng (Object Oriented Programming – OOP) là một kỹ thuật lập trình cho phép lập trình viên tạo ra các đối tượng code trừu tượng hóa các đối tượng giống với tự nhiên, đóng vai trò quan trọng và sử dụng nhiều hiện nay. Hầu hết các ngôn ngữ lập trình hiện nay như Java, PHP, .NET, Ruby, Python... đều hỗ trợ OOP.

1.2 Đối tượng

Một đối tượng (Object) bao gồm 2 thông tin: thuộc tính và phương thức.

- Thuộc tính (Attribute): là những thông tin, đặc điểm của đối tượng. Ví dụ: con người có các đặc tính như mắt, mũi, tay, chân...
- Phương thức (Method): là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, ...

1.3 Lớp

Một lớp là một kiểu dữ liệu bao gồm các thuộc tính và các phương thức được định nghĩa từ trước. Đây là sự trừu tượng hóa của đối tượng. Khác với kiểu dữ liệu thông thường, một lớp là một đơn vị (trừu tượng) bao gồm sự kết hợp giữa các phương thức và các thuộc tính. Hiểu nôm na hơn là các đối tượng có các đặc tính tương tự nhau được gom lại thành một lớp đối tượng.

2. CƠ SỞ LÝ THUYẾT

2.1 Tính đóng gói

Tính đóng gói (Encapsulation) là kỹ thuật làm cho các field trong một lớp thành private và cung cấp truy cập đến field đó thông qua public method. Các dữ liệu và phương thức có liên quan với nhau được đóng gói thành các lớp để tiện cho việc quản lý và sử dụng. Tức là mỗi lớp được xây dựng để thực hiện một nhóm

chức năng đặc trưng của riêng lớp đó. Ngoài ra, đóng gói còn để che giấu một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không thể nhìn thấy.

2.2 Tính kế thừa

Tính kế thừa (Inheritance) cho phép xây dựng một lớp mới dựa trên các định nghĩa của lớp đã có. Có nghĩa là lớp cha có thể chia sẻ dữ liệu và phương thức cho các lớp con. Các lớp con khỏi phải định nghĩa lại, ngoài ra có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới. Tái sử dụng mã nguồn 1 cách tối ưu, tận dụng được mã nguồn. Một số loại kế thừa thường gặp: đơn kế thừa, đa kế thừa, kế thừa đa cấp, kế thừa thứ bậc.

2.3 Tính đa hình

Tính đa hình (Polymorphism) là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau. Đây lại là một tính chất có thể nói là chứa đựng hầu hết sức mạnh của lập trình hướng đối tượng.

2.4 Tính trừu tượng

Tính trừu tượng (Abstraction) là tổng quát hóa một cái gì đó lên, không cần chú ý chi tiết bên trong. Nó không màng đến chi tiết bên trong là gì và người ta vẫn hiểu nó mỗi khi nghe về nó. Chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán đang lập trình. Vì một đối tượng có rất nhiều thuộc tính phương thức, nhưng với bài toán cụ thể không nhất thiết phải chọn tất cả.

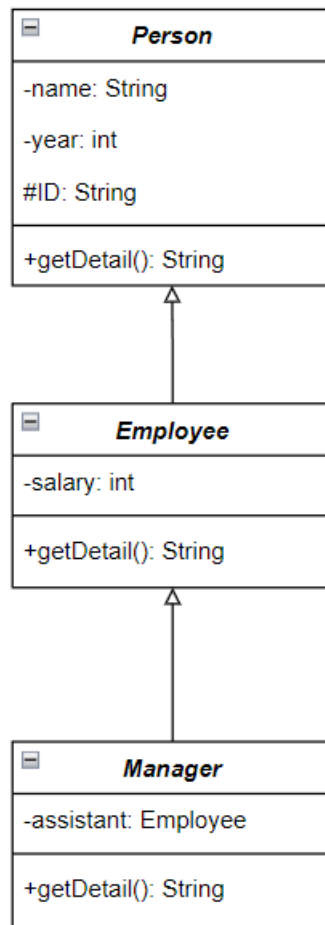
3. NỘI DUNG ÁP DỤNG VÀ PHƯƠNG PHÁP ÁP DỤNG

3.1 Nội dung áp dụng

Sử dụng ngôn ngữ java, các thư viện có sẵn kết hợp với kỹ thuật lập trình hướng đối tượng để tận dụng tính đóng gói, tính kế thừa, tính đa hình, tính trừu tượng áp dụng vào các bài toán thực tế được đưa ra.

3.1.1 Mô tả bài toán

- Bài toán 1:
 - Xây dựng lớp hình tròn với thuộc tính là bán kính, viết các hàm tính chu vi, diện tích của nó với độ dài bán kính người sử dụng nhập vào.
 - Xây dựng lớp hình trụ thừa kế lớp hình tròn với thuộc tính là chiều cao.
 - Viết chương trình tính diện tích xung quanh, diện tích đáy (2 đáy) hình trụ với chiều cao người sử dụng nhập vào.
- Bài toán 2:
 - Cho sơ đồ lớp sau:



- Cài đặt 3 lớp như trên biểu đồ (Tự tạo phương thức getter và setter ít nhất có thể, sao cho đảm bảo nguyên lý đóng gói & che dấu dữ liệu. Tạo các phương thức khởi dựng ít nhất có thể)
- Cài đặt lớp Test:
 - Nhập vào một công ty gồm ba mảng: 20 Person, 10 Employee, 5 Manager (đầy đủ dữ liệu). Khi nhập dữ liệu cho Manager, lưu ý assistant phải nằm trong số 10 Employee nói trên.
 - Tính xem có bao nhiêu người làm trợ lý (assistant) trong công ty, in ra người có lương cao nhất.
 - Nhập vào 1 Employee, kiểm tra xem trong công ty đã có Employee này chưa.
 - Tính tổng lương toàn công ty, in ra người có lương cao nhất.
- Bài toán 3:

- Một vũ trường mỗi tối thứ bảy có tổ chức khiêu vũ cho các cặp nam nữ. Cặp đầu tiên là người nam đến đầu tiên và người nữ đến đầu tiên, cặp thứ hai là người nam đến thứ hai và người nữ đến thứ hai,...
- Viết chương trình nhập những người đi dự khiêu vũ (tên và giới tính) và lưu vào trong collection (sử dụng Queue).
- Sau đó xuất ra màn hình tung cặp khiêu vũ và những người còn dư.

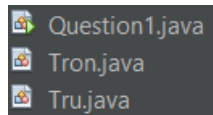
3.1.2 Ý tưởng xử lý bài toán

- Bài toán 1:
 - Sử dụng công thức hình tròn chu vi : $2\pi r$, diện tích: πr^2 .
 - Sử dụng kiến thức lập trình hướng đối tượng về tính kế thừa.
 - Sử dụng công thức hình trụ diện tích xung quanh: $2\pi rh$, diện tích toàn phần: $2 \times \text{diện tích hình tròn} + \text{diện tích xung quanh}$.
- Bài toán 2:
 - Sử dụng kiến thức lập trình hướng đối tượng về tính đóng gói, tính đa hình, tính kế thừa.
 - Sử dụng collection dạng mảng của java để lưu trữ 3 mảng.
 - Sử dụng collection chỉ lưu trữ các giá trị duy nhất của java lưu trữ từng thuộc assistant từ mảng Manager sau đó tính kích thước collection đó sẽ giải quyết được bài toán.
 - Duyệt qua các mảng Employee, Manager lần lượt công thêm thuộc tính salary, lấy đối tượng nào nếu có salary lớn hơn salary được lưu hiện tại.
- Bài toán 3:
 - Sử dụng 2 collection Queue để lưu trữ tương ứng giới tính nam và nữ được nhập vào.
 - Với mỗi cặp được xuất lấy 1 nam, 1 nữ từ 2 Queue sau đến khi 1 trong 2 Queue hết phần tử thì sẽ xuất các phần tử của Queue không rỗng cũng là những người còn dư.

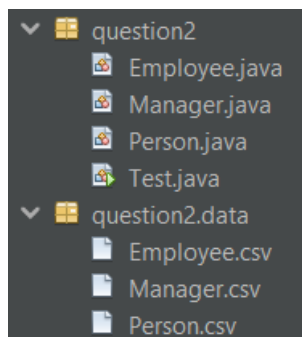
3.2 Phương pháp áp dụng

3.2.1 Cấu trúc thiết kế trong bài toán


- Bài toán 1: bài toán được tạo 2 lớp Tron, Tru, 1 file java chứa main



- Bài toán 2: bài toán này được nhập dữ liệu bằng các đọc các file csv tương ứng với 3 mảng Employee, Manager, Person. Với mỗi dòng trong file csv chứa đầy đủ thông tin của lớp tương ứng. Có 3 lớp chính: Employee, Manager, Person và 1 lớp Test chứa main.




- Employee.csv: 4 thuộc tính name, year, ID, salary (USA)

 Employee.csv - Notepad

```
File Edit Format View Help
name,year,ID,salary (USA)
Phạm Anh Huy,2001,21,218
Đoàn Trung Kiên,2001,22,323
Trần Quang Kiên,2001,23,309
Nguyễn Tuấn Kiệt,2002,24,229
Huỳnh Thanh Kỳ,2001,25,279
Phạm Ngọc Khải,2001,26,231
Nguyễn Đỗ An Khang,2001,27,282
Nguyễn Huỳnh Dương Khang,2001,28,197
Đào Bảo Khanh,2001,29,189
Trương Nguyễn Duy Khiêm,2001,30,188
Nguyễn Anh Khoa,2001,31,285
Trần Anh Khoa,2000,32,260
Nguyễn Đình Khôi,2001,33,112
Lê Mỹ Thanh Lành,2001,34,154
Nguyễn Bảo Linh,2000,35,222
|
```


- Person.csv: có 3 thuộc tính name, year, ID

 Person.csv - Notepad

File Edit Format View Help

```
name,year,ID
Huỳnh Bảo Thế Anh,2001,1
Đoàn Hoàng Ca,2000,2
Nông Minh Cảnh,2001,3
Phạm Đào Cao,2000,4
Nguyễn Quốc Dũng,2001,5
Lê Quốc Đạt,2001,6
Nguyễn Quốc Đạt,2001,7
Trần Văn Điệp,2001,8
Hồ Minh Đức,2001,9
Lê Công Hoài Đức,2001,10
Tăng Thanh Đức,2001,11
Trần Hoài Đức,2001,12
Nguyễn Quang Hải,2001,13
Đặng Thái Hiệp,2000,14
Trương Thanh Hiếu,2001,15
Ngô Nhật Hoàng,2001,16
Nguyễn Minh Hoàng,2001,17
Vũ Đức Hoàng,2001,18
Dương Việt Huy,2001,19
Ngô Quang Huy,2001,20
```

- Manager.csv: có 5 thuộc tính name, year, ID, salary (USA), assistant (ID)

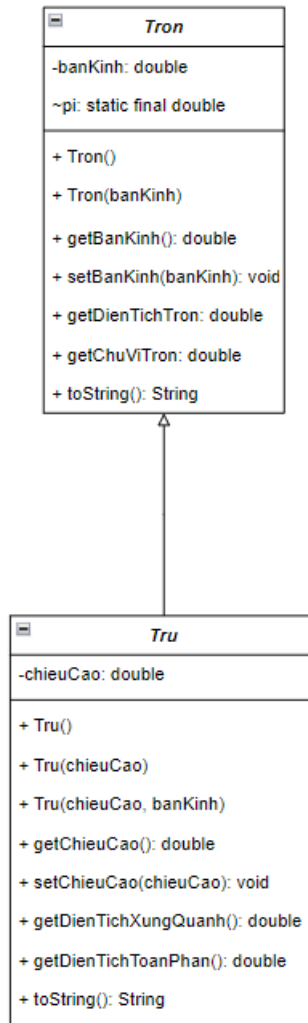
 Manager.csv - Notepad

File Edit Format View Help

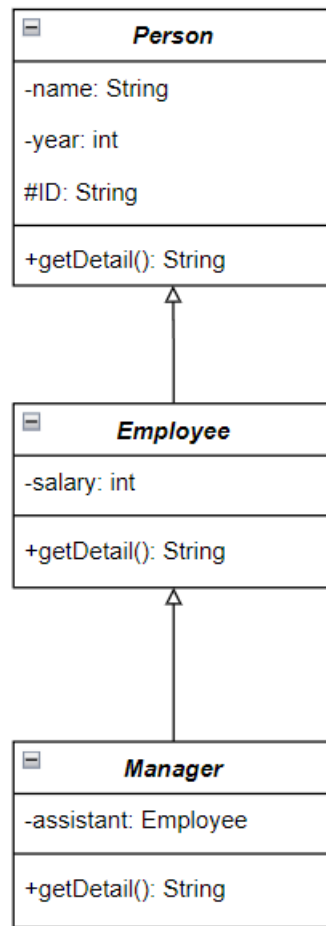
```
name,year,ID,salary (USA),assistant (ID)
Đỗ Văn Long,2001,36,23308,23
Huỳnh Võ Hoàng Long,2001,37,13763,22
Phạm Đình Hải Long,2001,38,22521,31
Đặng Hữu Lộc,2001,39,25379,23
Phan Tuấn Nam,2000,40,12657,26
```


3.2.2 Sơ đồ lớp

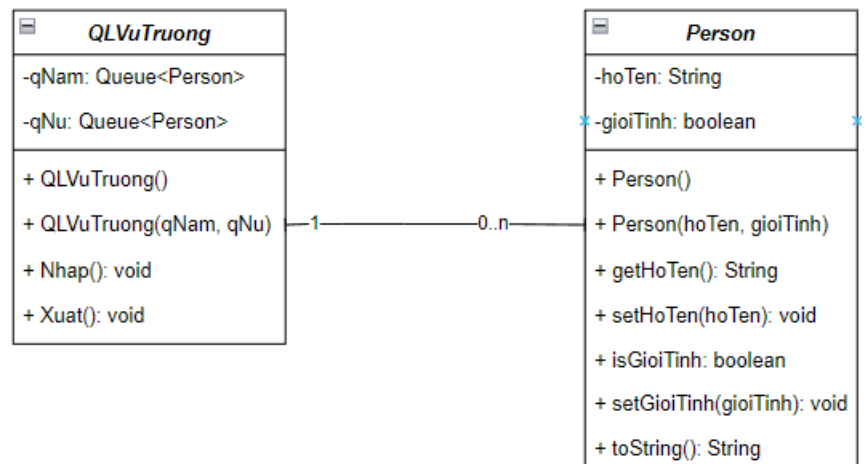
- Bài toán 1:



- Bài toán 2:



- Bài toán 3:



3.2.3 Code

- Bài toán 1:
 - Question1.java:

```
package question1;
```

```
/**
```

```
*
```

```
* @author tansi
```

```
*/
```

```
public class Question1 {
```

```
/**
```

```
* @param args the command line arguments
```

```
*/
```

```
public static void main(String[] args) {
```

```
// TODO code application logic here
```

```
Tron t1 = new Tron(0.5);
```

```
System.out.println(t1.toString());
```

```
Tru t2 = new Tru(0.5,0.5);
```

```
System.out.println(t2.toString());
```

```
}
```

```
}
```

- Tron.java:

```
package question1;
```

```
/**
```

```
*
```

```
* @author tansi
```

```
*/
```

```
public class Tron {
```

```
private double banKinh;  
static final double pi = 3.14;
```

```
public Tron() {  
    banKinh = 0;  
}
```

```
public Tron(double banKinh) {  
    if (banKinh >= 0)  
        this.banKinh = banKinh;  
    else  
        System.out.println("banKinh phải >= 0");  
}
```

```
public double getBanKinh() {  
    return banKinh;  
}
```

```
public void setBanKinh(double banKinh) {  
    this.banKinh = banKinh;  
}
```

```
public double getDienTichTron(){  
    return pi*banKinh*banKinh;  
}
```

```
public double getChuViTron(){  
    return pi*2*banKinh;  
}
```

```
@Override
```

```

    public String toString() {
        return "Tron{" +
            "Bán kính=" + banKinh +
            ", Chu vi hình tròn=" + getChuViTron() +
            ", Diện tích hình tròn=" + getDienTichTron() +
            '}';
    }
}

```

○ Tru.java:

```

package question1;

/**
 *
 * @author tansi
 */
public class Tru extends Tron{
    private double chieuCao;

    public Tru() {
    }

    public Tru(double chieuCao) {
        this.chieuCao = chieuCao;
    }

    public Tru(double chieuCao, double banKinh) {
        super(banKinh);
        if (chieuCao >= 0)
            this.chieuCao = chieuCao;
    }
}

```

```

        else
            System.out.println("Chiều cao phải >= 0");
    }

    public double getChieuCao() {
        return chieuCao;
    }

    public void setChieuCao(double chieuCao) {
        this.chieuCao = chieuCao;
    }

    public double getDienTichXungQuanh(){
        return 2*pi*getBanKinh()*chieuCao;
    }

    public double getDienTichToanPhan(){
        return 2*getDienTichTron()+getDienTichXungQuanh();
    }

    @Override
    public String toString() {
        return "Tru{" +
            "Bán kính đáy=" + getBanKinh() +
            ", Chiều cao=" + chieuCao +
            ", Diện tích xung quanh=" + getDienTichXungQuanh() +
            ", Diện tích toàn phần=" + getDienTichToanPhan() +
            '}';
    }

}

```

- Bài toán 2:
 - Person.java:

```
package question2;
```

```
/**
```

```
*
```

```
* @author tansi
```

```
*/
```

```
public class Person {
```

```
    private String name;
```

```
    private int year;
```

```
    protected String ID;
```

```
public Person() {
```

```
}
```

```
public Person(String name, int year, String ID) {
```

```
    this.name = name;
```

```
    this.year = year;
```

```
    this.ID = ID;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;
```

```
}
```

```
public int getYear() {
```

```

        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getDetail() {
        return "Person{" +
            "ID=" + ID +
            ", name=" + name +
            ", year=" + year +
            '}';
    }

}

```

○ Employee.java:

```
package question2;
```

```
/**
```

```
*
```

```
* @author tansi
```


***/**

```
public class Employee extends Person {  
    private int salary;  
  
    public Employee() {  
    }  
  
    public Employee(int salary) {  
        this.salary = salary;  
    }  
  
    public Employee(int salary, String name, int year, String ID) {  
        super(name, year, ID);  
        this.salary = salary;  
    }  
  
    public int getSalary() {  
        return salary;  
    }  
  
    public void setSalary(int salary) {  
        this.salary = salary;  
    }  
  
    public String getID() {  
        return ID;  
    }  
  
    public void setID(String ID) {  
        this.ID = ID;  
    }
```

@Override

```
public String getDetail() {  
    return "Employee{" +  
        "ID=" + ID +  
        ", name=" + getName() +  
        ", year=" + getYear() +  
        ", salary=" + salary +  
        '}';  
}  
  
}
```

○ Manager.java:

package question2;

```
/**  
*  
* @author tansi  
*/  
  
public class Manager extends Employee {  
    private Employee assistant;  
  
    public Manager() {  
}  
  
    public Manager(Employee assistant) {  
        this.assistant = assistant;  
}
```

```

    public Employee getAssistant() {
        return assistant;
    }

    public void setAssistant(Employee assistant) {
        this.assistant = assistant;
    }
    @Override
    public String getDetail() {
        return "Manager{" +
            "ID=" + ID +
            ", name=" + getName() +
            ", year=" + getYear() +
            ", salary=" + getSalary() +
            ", assistant=" + assistant.getDetail() +
            '}';
    }
}

```

○ Test.java:

```

package question2;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Scanner;

/**
 *

```

*** @author tansi**

***/**

public class Test {

/**

*** @param args the command line arguments**

***/**

public static void main(String[] args) {

// TODO code application logic here

Scanner sc = new Scanner(System.in);

String path_filePerson = "src\\question2\\data\\Person.csv";

ArrayList<Person> arrPerson = pArr(path_filePerson);

printArrayPerson(arrPerson);

String path_fileEmployee = "src\\question2\\data\\Employee.csv";

ArrayList<Employee> arrEmployee = eArr(path_fileEmployee);

printArrayEmployee(arrEmployee);

String path_fileManager = "src\\question2\\data\\Manager.csv";

ArrayList<Manager> arrManager = mArr(path_fileManager, arrEmployee);

printArrayManager(arrManager);

System.out.println();

System.out.println("Số trợ lý trong công ty: "+countAssistant(arrManager));

System.out.println("Nhập id Employee: ");

String id = sc.next();

checkEmployee(id, arrEmployee);

```

        statisticsSalary(arrEmployee, arrManager);

    }

    public static ArrayList<Person> pArr(String path_filePerson){
        String file = path_filePerson;
        BufferedReader reader = null;
        String line = "";
        ArrayList<Person> arrayPerson = new ArrayList<Person>();

        try{
            reader = new BufferedReader(new FileReader(file));
            line = reader.readLine();
            while((line = reader.readLine())!=null){
                Person p = new Person();
                int i_row = 0;
                String[] row = line.split(",");
                for(String index: row){
                    if(i_row == 0){
                        p.setName(index);
                    }
                    else if (i_row == 1){
                        int id = Integer.parseInt(index);
                        p.setYear(id);
                    }
                    else{
                        p.setID(index);
                    }
                    i_row+=1;
                }
                arrayPerson.add(p);
            }
        }
    }

```

```

    }
} catch (Exception e) {
    e.printStackTrace();
}

finally {
    try {
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
return arrayPerson;
}

```

```

public static ArrayList<Employee> eArr(String path_fileEmployee) {
    String file = path_fileEmployee;
    BufferedReader reader = null;
    String line = "";
    ArrayList<Employee> arrayEmployee = new ArrayList<Employee>();

    try {
        reader = new BufferedReader(new FileReader(file));
        line = reader.readLine();
        while ((line = reader.readLine()) != null) {
            Employee e = new Employee();
            int i_row = 0;
            String[] row = line.split(",");
            for (String index: row) {
                if (i_row == 0) {
                    e.setName(index);

```

```

    }
    else if (i_row == 1){
        int id = Integer.parseInt(index);
        e.setYear(id);
    }
    else if (i_row == 2){
        e.setID(index);
    }
    else{
        int salary = Integer.parseInt(index);
        e.setSalary(salary);
    }
    i_row+=1;
}
arrayEmployee.add(e);
}
}catch(Exception e){
    e.printStackTrace();
}

finally{
    try{
        reader.close();
    }catch(IOException e){
        e.printStackTrace();
    }
}
return arrayEmployee;
}

```

```

public static Employee findEmployee (String ID, ArrayList<Employee> eArr){

```

```

    for (int i=0;i<eArr.size();i++){
        if(eArr.get(i).getID().equals(ID))
            return eArr.get(i);
    }
    return null;
}

```

```

    public static ArrayList<Manager> mArr(String path_fileManager,
    ArrayList<Employee> eArr){
        String file = path_fileManager;
        BufferedReader reader = null;
        String line = "";
        ArrayList<Manager> arrayManager = new ArrayList<Manager>();

        try{
            reader = new BufferedReader(new FileReader(file));
            line = reader.readLine();
            while((line = reader.readLine())!=null){
                Manager m = new Manager();
                int i_row = 0;
                String[] row = line.split(",");
                for(String index: row){
                    if(i_row == 0){
                        m.setName(index);
                    }
                    else if (i_row == 1){
                        int id = Integer.parseInt(index);
                        m.setYear(id);
                    }
                    else if (i_row == 2){
                        m.setID(index);

```



```

        }
        else if (i_row == 3){
            int salary = Integer.parseInt(index);
            m.setSalary(salary);
        }
        else{
            Employee e = findEmployee(index, eArr);
            m.setAssistant(e);
        }
        i_row+=1;
    }
    arrayManager.add(m);
}
}catch(Exception e){
    e.printStackTrace();
}

finally{
    try{
        reader.close();
    }catch(IOException e){
        e.printStackTrace();
    }
}
return arrayManager;
}

public static void printArrayPerson(ArrayList<Person> arr){
    System.out.println("Danh sách Person: ");
    for(int i=0;i<arr.size();i++){
        System.out.println(arr.get(i).getDetail());
    }
}

```

```
    }  
    System.out.println();  
}
```

```
public static void printArrayEmployee(ArrayList<Employee> arr){  
    System.out.println("Danh sách Employee: ");  
    for(int i=0;i<arr.size();i++){  
        System.out.println(arr.get(i).getDetail());  
    }  
    System.out.println();  
}
```

```
public static void printArrayManager(ArrayList<Manager> arr){  
    System.out.println("Danh sách Manager: ");  
    for(int i=0;i<arr.size();i++){  
        System.out.println(arr.get(i).getDetail());  
    }  
    System.out.println();  
}
```

```
public static int countAssistant(ArrayList<Manager> arr){  
    HashSet<String> idAssistant = new HashSet<>();  
    for (int i=0;i<arr.size();i++){  
        Employee assistant = arr.get(i).getAssistant();  
        if (assistant!=null)  
            idAssistant.add(assistant.getID());  
    }  
    return idAssistant.size();  
}
```

```
public static void checkEmployee(String id, ArrayList<Employee> arr){
```

```

    Employee e = findEmployee(id, arr);
    if (e!=null)
        System.out.println("Đã tồn tại: " + e.getDetail());
    else
        System.out.println("Chưa có Employee này!!");
}

    public static void statisticsSalary(ArrayList<Employee> eArr,
    ArrayList<Manager> mArr){
        Employee e = eArr.get(0);
        int sumSalary = 0;

        for (int i=0;i<eArr.size();i++){
            sumSalary+=eArr.get(i).getSalary();
            if (e.getSalary()<eArr.get(i).getSalary())
                e = eArr.get(i);
        }

        for (int i=0;i<mArr.size();i++){
            sumSalary+=mArr.get(i).getSalary();
            if (e.getSalary()<mArr.get(i).getSalary())
                e = mArr.get(i);
        }

        System.out.println("Tổng số lương công ty: "+sumSalary+" USD");
        System.out.println("Người có lương cao nhất: " + e.getDetail());

    }

}

```

- Bài toán 3:
 - Person.java:

```
package question3;
```

```
/**
```

```
*
```

```
* @author tansi
```

```
*/
```

```
public class Person {
```

```
    private String hoTen;
```

```
    private boolean gioiTinh;
```

```
    public Person() {
```

```
    }
```

```
    public Person(String hoTen, boolean gioiTinh) {
```

```
        this.hoTen = hoTen;
```

```
        this.gioiTinh = gioiTinh;
```

```
    }
```

```
    public String getHoTen() {
```

```
        return hoTen;
```

```
    }
```

```
    public void setHoTen(String hoTen) {
```

```
        this.hoTen = hoTen;
```

```
    }
```

```
    public boolean isGioiTinh() {
```

```
        return gioiTinh;
```

```
    }
```

```

    public void setGioiTinh(boolean gioiTinh) {
        this.gioiTinh = gioiTinh;
    }

    @Override
    public String toString() {
        if (gioiTinh==true)
            return "Person{" + "hoTen=" + hoTen + ", gioiTinh=" + "Nam" + '}';
        return "Person{" + "hoTen=" + hoTen + ", gioiTinh=" + "Nu" + '}';
    }

}

```

○ QLVuTruong.java:

```

package question3;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

/**
 *
 * @author tansi
 */
public class QLVuTruong {
    static Scanner sc = new Scanner(System.in);

    private Queue<Person> qNam;
    private Queue<Person> qNu;

```

```
public QLVuTruong() {  
    this.qNam = new LinkedList();  
    this.qNu = new LinkedList();  
}
```

```
public QLVuTruong(Queue<Person> qNam, Queue<Person> qNu) {  
    this.qNam = qNam;  
    this.qNu = qNu;  
}
```

```
public void Nhap(){  
    while (true){  
        System.out.println("Quản Lý Vũ Trường");  
        System.out.println("Enter 1: Đăng ký khiêu vũ");  
        System.out.println("Enter 2: Để Thoát");  
        String line = sc.nextLine();  
        switch(line){  
            case "1":{  
                System.out.print("Nhập Họ Tên:");  
                String hoten = sc.nextLine();  
                System.out.print("Bạn có phải Nam không ? True/False ");  
                boolean gioitinh = sc.nextBoolean();  
                Person p = new Person(hoten,gioitinh);  
                if (gioitinh==true)  
                    qNam.add(p);  
                else  
                    qNu.add(p);  
                break;  
            }  
            case "2":{  
                return;  
            }  
        }  
    }  
}
```

```

    }
    default:{
        System.out.print("Vui lòng nhập đúng Lựa Chọn!!");
        continue;
    }
}
}
}
}
}

```

```

public void Xuat(){
    int n_cap = Math.min(qNam.size(), qNu.size());
    System.out.println("Danh sách cặp khiêu vũ:");
    for(int i=0;i<n_cap;i++){
        if (!qNam.isEmpty() && !qNu.isEmpty()){
            System.out.println(qNam.remove().toString()+"
Cặp
"+qNu.remove().toString());
        }
    }
    int n_du = Math.max(qNam.size(), qNu.size());
    System.out.println("Danh sách những người chưa có cặp:");
    if (n_du==qNam.size()){
        System.out.println(qNam);
    }else{
        System.out.println(qNu);
    }
}
}
}

```

- Question3.java:

```
package question3;
```

```
/**
```

```
*
```

```
* @author tansi
```

```
*/
```

```
public class Question3 {
```

```
/**
```

```
* @param args the command line arguments
```

```
*/
```

```
public static void main(String[] args) {
```

```
// TODO code application logic here
```

```
QLVuTruong ql = new QLVuTruong();
```

```
ql.Nhap();
```

```
ql.Xuat();
```

```
}
```

```
}
```

3.2.4 Xuất kết quả

- Bài toán 1:

```
run:
Tron(Bán kính=0.5, Chu vi hình tròn=3.14, Diện tích hình tròn=0.785)
Tru(Bán kính đáy=0.5, Chiều cao=0.5, Diện tích xung quanh=1.57, Diện tích toàn phần=3.14)
BUILD SUCCESSFUL (total time: 0 seconds)
```


- Bài toán 2:

```
Output - Question2 (run) X
run:
Danh sách Person:
Person(ID=1, name=Huỳnh Bảo Thế Anh, year=2001)
Person(ID=2, name=Đoàn Hoàng Ca, year=2000)
Person(ID=3, name=Nông Minh Cảnh, year=2001)
Person(ID=4, name=Phạm Đào Cao, year=2000)
Person(ID=5, name=Nguyễn Quốc Dũng, year=2001)
Person(ID=6, name=Lê Quốc Đạt, year=2001)
Person(ID=7, name=Nguyễn Quốc Đạt, year=2001)
Person(ID=8, name=Trần Văn Điệp, year=2001)
Person(ID=9, name=Hồ Minh Đức, year=2001)
Person(ID=10, name=Lê Công Hoài Đức, year=2001)
Person(ID=11, name=Tăng Thanh Đức, year=2001)
Person(ID=12, name=Trần Hoài Đức, year=2001)
Person(ID=13, name=Nguyễn Quang Hải, year=2001)
Person(ID=14, name=Đặng Thái Hiệp, year=2000)
Person(ID=15, name=Trương Thanh Hiếu, year=2001)
Person(ID=16, name=Ngô Nhật Hoàng, year=2001)
Person(ID=17, name=Nguyễn Minh Hoàng, year=2001)
Person(ID=18, name=Vũ Đức Hoàng, year=2001)
Person(ID=19, name=Dương Việt Huy, year=2001)
Person(ID=20, name=Ngô Quang Huy, year=2001)
```

```
Output - Question2 (run) X
Danh sách Employee:
Employee(ID=21, name=Phạm Anh Huy, year=2001, salary=218)
Employee(ID=22, name=Đoàn Trung Kiên, year=2001, salary=323)
Employee(ID=23, name=Trần Quang Kiên, year=2001, salary=309)
Employee(ID=24, name=Nguyễn Tuấn Kiệt, year=2002, salary=229)
Employee(ID=25, name=Huỳnh Thanh Kỳ, year=2001, salary=279)
Employee(ID=26, name=Phạm Ngọc Khải, year=2001, salary=231)
Employee(ID=27, name=Nguyễn Đỗ An Khang, year=2001, salary=282)
Employee(ID=28, name=Nguyễn Huỳnh Dương Khang, year=2001, salary=197)
Employee(ID=29, name=Đào Bảo Khanh, year=2001, salary=189)
Employee(ID=30, name=Trương Nguyễn Duy Khiêm, year=2001, salary=188)
Employee(ID=31, name=Nguyễn Anh Khoa, year=2001, salary=285)
Employee(ID=32, name=Trần Anh Khoa, year=2000, salary=260)
Employee(ID=33, name=Nguyễn Đình Khối, year=2001, salary=112)
Employee(ID=34, name=Lê Mỹ Thanh Lành, year=2001, salary=154)
Employee(ID=35, name=Nguyễn Bảo Linh, year=2000, salary=222)

Danh sách Manager:
Manager(ID=36, name=Đỗ Văn Long, year=2001, salary=23308, assistant=Employee(ID=23, name=Trần Quang Kiên, year=2001, salary=309))
Manager(ID=37, name=Huỳnh Võ Hoàng Long, year=2001, salary=13763, assistant=Employee(ID=22, name=Đoàn Trung Kiên, year=2001, salary=323))
Manager(ID=38, name=Phạm Đình Hải Long, year=2001, salary=22521, assistant=Employee(ID=31, name=Nguyễn Anh Khoa, year=2001, salary=285))
Manager(ID=39, name=Đặng Hữu Lộc, year=2001, salary=25379, assistant=Employee(ID=23, name=Trần Quang Kiên, year=2001, salary=309))
Manager(ID=40, name=Phan Tuấn Nam, year=2000, salary=12657, assistant=Employee(ID=26, name=Phạm Ngọc Khải, year=2001, salary=231))

Số trợ lý trong công ty: 4
Nhập id Employee:
23
Đã tồn tại: Employee(ID=23, name=Trần Quang Kiên, year=2001, salary=309)
Tổng số lương công ty: 101106 USD
Người có lương cao nhất: Manager(ID=39, name=Đặng Hữu Lộc, year=2001, salary=25379, assistant=Employee(ID=23, name=Trần Quang Kiên, year=2001, salary=309))
BUILD SUCCESSFUL (total time: 4 seconds)
```

- Bài toán 3:

```
Quản Lý Vũ Trường
Enter 1: Đăng ký khiêu vũ
Enter 2: Để Thoát
1
Nhập Họ Tên:
Pham Minh Tuan
Bạn có phải Nam không ? True/False
True
```

```
Quản Lý Vũ Trường
Enter 1: Đăng ký khiêu vũ
Enter 2: Để Thoát
1
Nhập Họ Tên:
Pham Minh A
Bạn có phải Nam không ? True/False
False
```

```
Quản Lý Vũ Trường
Enter 1: Đăng ký khiêu vũ
Enter 2: Để Thoát
1
Nhập Họ Tên:
Pham Minh B
Bạn có phải Nam không ? True/False
True
```

```
Quản Lý Vũ Trường
Enter 1: Đăng ký khiêu vũ
Enter 2: Để Thoát
2
Danh sách cặp khiêu vũ:
Person(hoTen=Pham Minh Tuan, gioiTinh=Nam) Cặp Person(hoTen=Pham Minh A, gioiTinh=Nu)
Danh sách những người chưa có cặp:
[Person(hoTen=Pham Minh B, gioiTinh=Nam)]
BUILD SUCCESSFUL (total time: 57 seconds)
```

4. KẾT LUẬN

Dựa trên nguyên lý kế thừa, trong quá trình mô tả các lớp có thể loại bỏ những chương trình bị lặp, dư. Và có thể mở rộng khả năng sử dụng các lớp mà không cần thực hiện lại. Tối ưu và tái sử dụng code hiệu quả. Đảm bảo rút ngắn thời gian xây dựng hệ thống và tăng năng suất thực hiện. Sự xuất hiện của 2 khái niệm mới là lớp và đối tượng chính là đặc trưng của phương pháp lập trình hướng đối tượng. Nó đã giải quyết được các khuyết điểm của phương pháp lập trình hướng cấu trúc để lại. Ngoài ra 2 khái niệm này đã giúp biểu diễn tốt hơn thế giới thực trên máy tính.

TÀI LIỆU THAM KHẢO

- [1] <https://topdev.vn/blog/oop-la-gi/>
- [2] <https://nhathauxaydung24h.com/dien-tich-hinh-tron>
- [3] https://vi.wikipedia.org/wiki/H%C3%ACnh_tr%E1%BB%A5_tr%C3%B2n
- [4] <https://niithanoi.edu.vn/tat-tan-tat-ve-queue-trong-java.html>
- [5] <https://viettuts.vn/java-collection/hashset-trong-java>
- [6] <https://gpcoder.com/3107-doc-ghi-file-csv-trong-java/>

.....