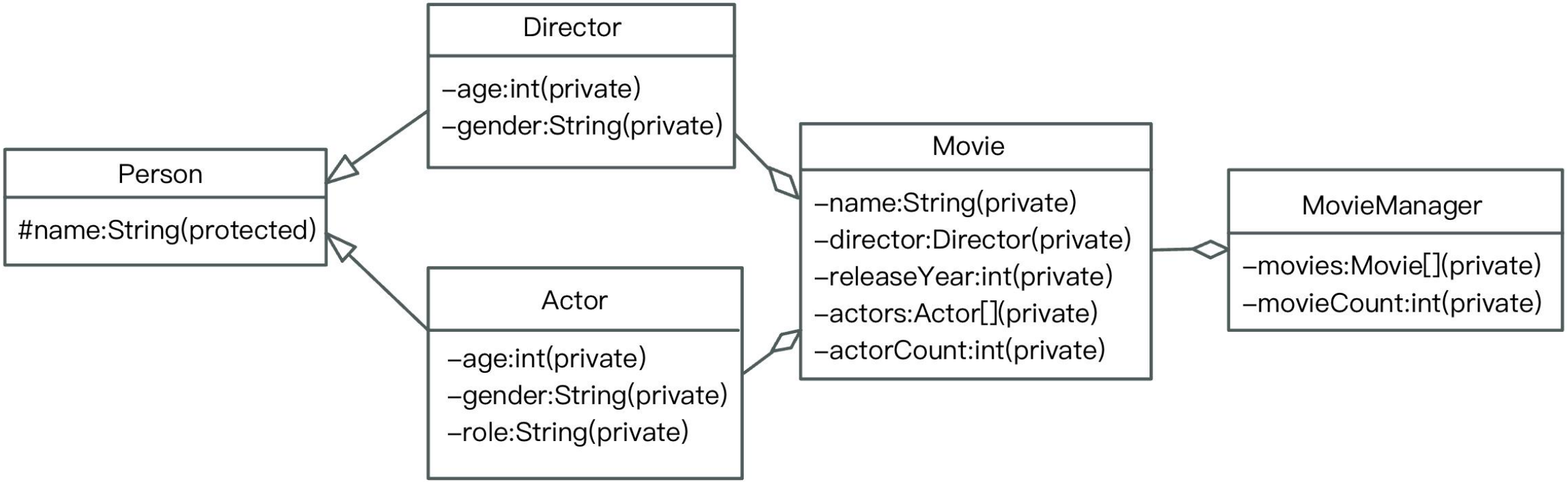
電影管理系統
Movie System

11011142曾鈺涵、11011209謝佳璇

class diagram

- **superclass ：人Person**
  - Person ： name(protect)
  - director：age(private) 、gender(private)
  - actors：age(private)、gender(private) 、role(private)
- **subclass：導演Director**
- **subclass：演員Actor**
- **Class：電影Movie**
  - (name、director、releaseYear、actors、actorCount) (private)
- **Class：電影管理MovieManager**
  - movies(private)、movieCount (private)

class diagram



系統功能和作用

主要功能是實現電影管理系統，通過物件導向設計讓使用者能夠建立電影、導演、演員的資料並管理多部電影。

1. 添加一部電影(名稱、導演、上映日期、演員人數、演員)
2. 根據電影名稱刪除電影
3. 依照電影名稱或導演搜索該電影
4. 顯示所有電影(這部分我們無法顯示出所有清單所以程式碼顯示不出結果)

物件導向觀念

- 類別與物件
- 繼承
- 封裝： Movie
- 多型： override
- 關聯性與組合
- 模組化：類別分工

類別與物件

```
public class MovieManager {

class Person {

class Director extends Person {

class Actor extends Person {

class Movie {
```

- 我們的系統中有5個類別:
  MovieManger、Person
  、Director、Actor、Movie

```
public class MovieManager {
    private Movie[] movies;
    private int movieCount;

    public MovieManager(int i) {

    public void addMovie(String name, Director director, int releaseYear, Actor[] actors) {

    public void deleteMovie(String name){

    public void searchMovieByName(String name){

    public void searchMovieByDirector(String directorName){

    public void displayAllMovies(){

    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        MovieManager manager = new MovieManager(100000);
```

- 物件是根據類別創建的實例，像是:
  MovieManger類別中的物件在main()中初始化了
  MovieManger manger = new MovieManger(100000);

繼承

- 好處：讓程式碼更具結構化，避免重複定義如 name 等屬性。
- Actor、Director繼承父類別Person，使用Person類別中定義的屬性和行為。

```
class Person{
    protected String name;

    public Person(String name){
        this.name = name;}

    public String getName(){
        return name;}
    }
```

Person裡面的name設為protected而非private是因為需要讓它的子類別Director和Actor直接存取name，不需要再使用getName()來取得名稱。

```java
class Director extends Person{
    private int age;
    private String gender;

    public Director(String name, int age, String gender){
        super(name);
        this.age = age;
        this.gender = gender;}

    @Override
    public String data(){
        return "導演: "+name+", 年齡: "+age+", 性別: "+gender;}
}
```

```java
class Actor extends Person{
    private int age;
    private String gender;
    private String role;

    public Actor(String name, int age, String gender, String role){
        super(name);
        this.age = age;
        this.gender = gender;
        this.role = role;
    }

    @Override
    public String data(){
        return "演員: " + name + ", 年齡: " + age + ", 性別: " + gender + ", 角色: " + role;}
}
```

封裝

將屬性設為 private，並透過 public 方法（例如 getName、getDirector 等）來訪問它們，這樣可以保護資料，防止外部直接修改。

```
class Movie{
    private String name;
    private Director director;
    private int releaseYear;
    private Actor[] actors;

    public Movie(String name, Director director, int releaseYear, Actor[] actors) {

    public String getName(){
        return name;
    }

    public Director getDirector(){
        return director;
    }
```

多型

- 覆寫（Override）父類別方法，如圖所示data方法。處理多種子類別時更加彈性。

```
class Person{
    protected String name;

    public Person(String name){
        this.name = name;}

    public String getName(){
        return name;}

    public String data(){
        return "Person{name='" + name + "'}";
}

class Director extends Person{
    private int age;
    private String gender;

    public Director(String name, int age, String gender){
        super(name);
        this.age = age;
        this.gender = gender;}

    @Override
    public String data(){
        return "導演: "+name+", 年齡: "+age+", 性別: "+gender;}
}
```

Person類別中有一個data()，他返回了一個字串。而Director類別繼承了Person類別，並且覆寫了data()方法。

關聯性

- Movie包含Actor、Director，顯示一部電影和一位導演與多名演員之間擁有關聯性。

```
class Movie{
    private String name;
    private Director director;
    private int releaseYear;
    private Actor[] actors;

    public Movie(String name, Director director, int releaseYear, Actor[] actors){
        this.name = name;
        this.director = director;
        this.releaseYear = releaseYear;
        this.actors = actors;
    }

    public String getName(){
        return name;
    }

    public Director getDirector(){
        return director;
    }
}
}
```

模組化

Person 提供基本屬性。

Director 和 Actor 提供導演與演員的詳細資訊。

Movie 管理單部電影的屬性和行為。

MovieManager 管理多部電影的集合操作（例如新增、刪除、查詢電影）。

Thanks for listening