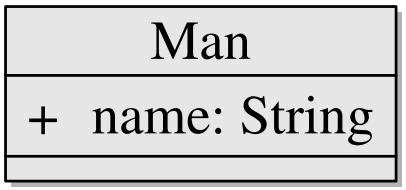
Clase Interne

Alexandru Olteanu

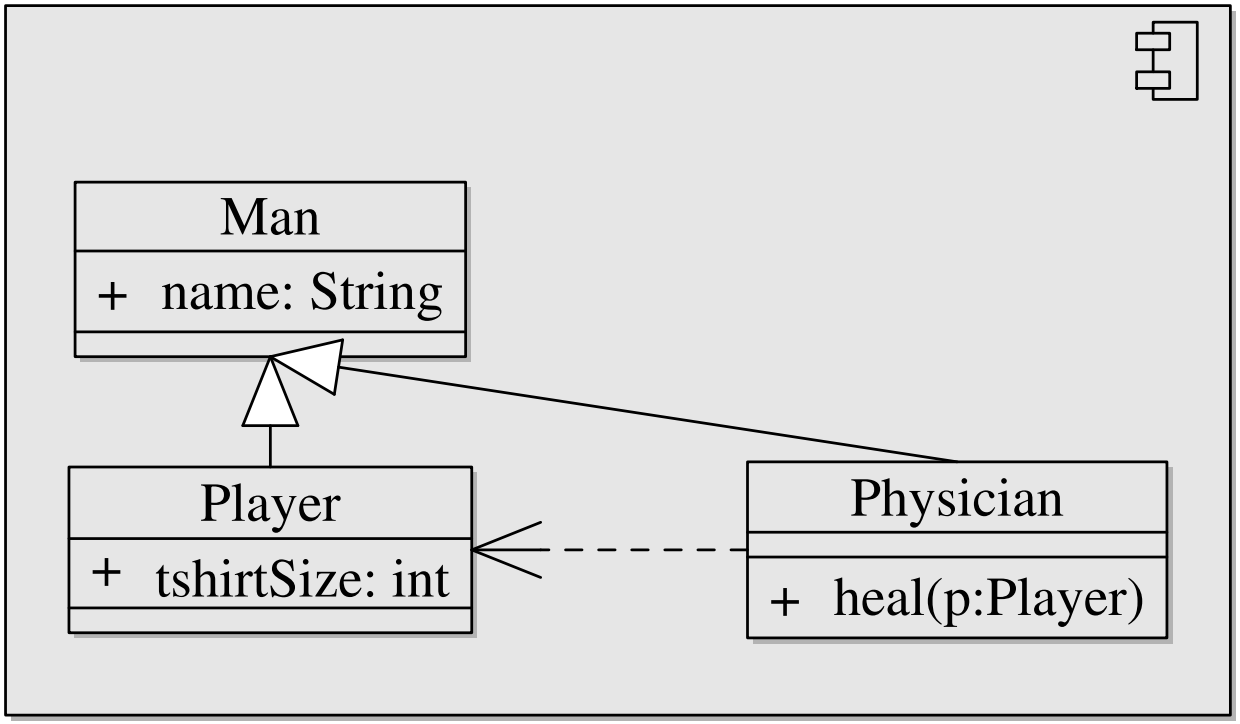
Universitatea Politehnica Bucuresti
Facultatea de Automatică si Calculatoare, Departamentul Calculatoare
alexandru.olteanu@upb.ro

OOP, 2020

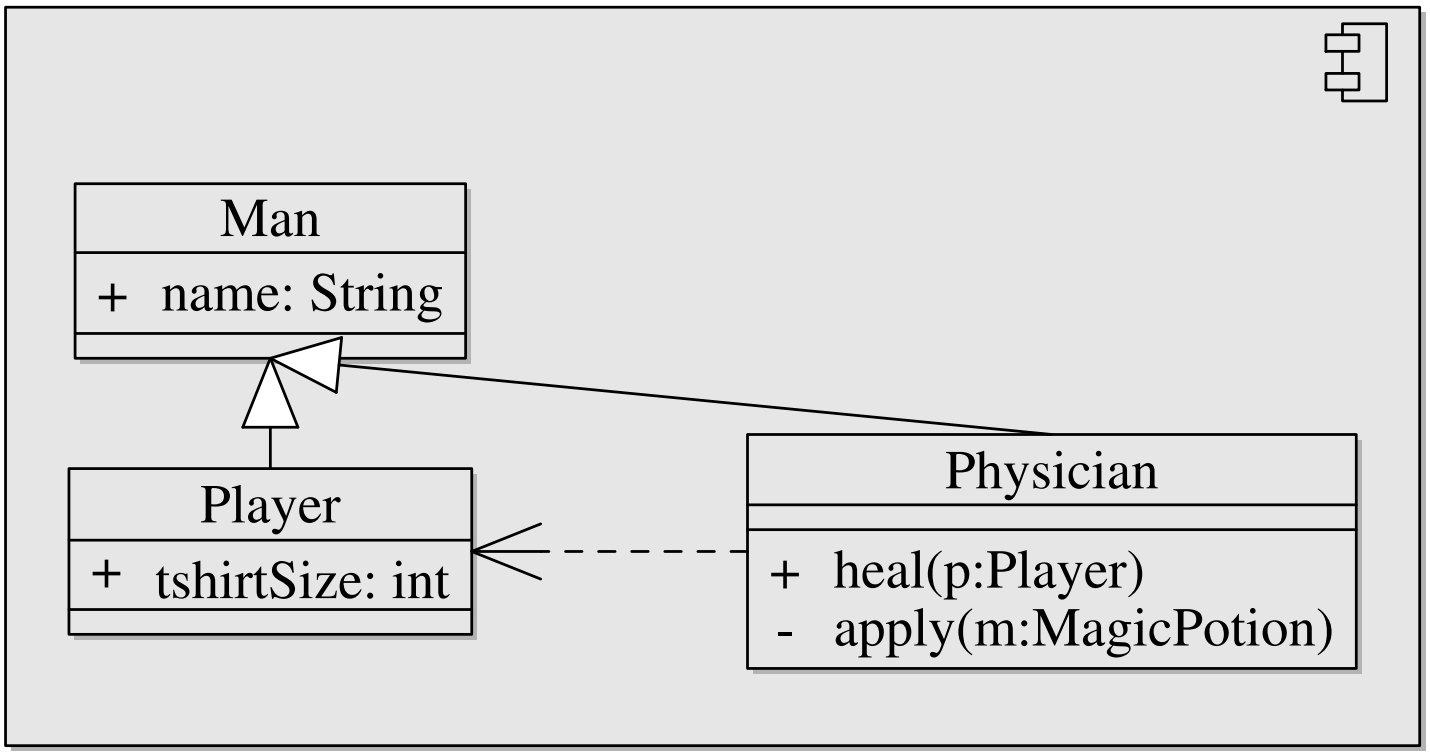
Curs 2: ce contine o clasa



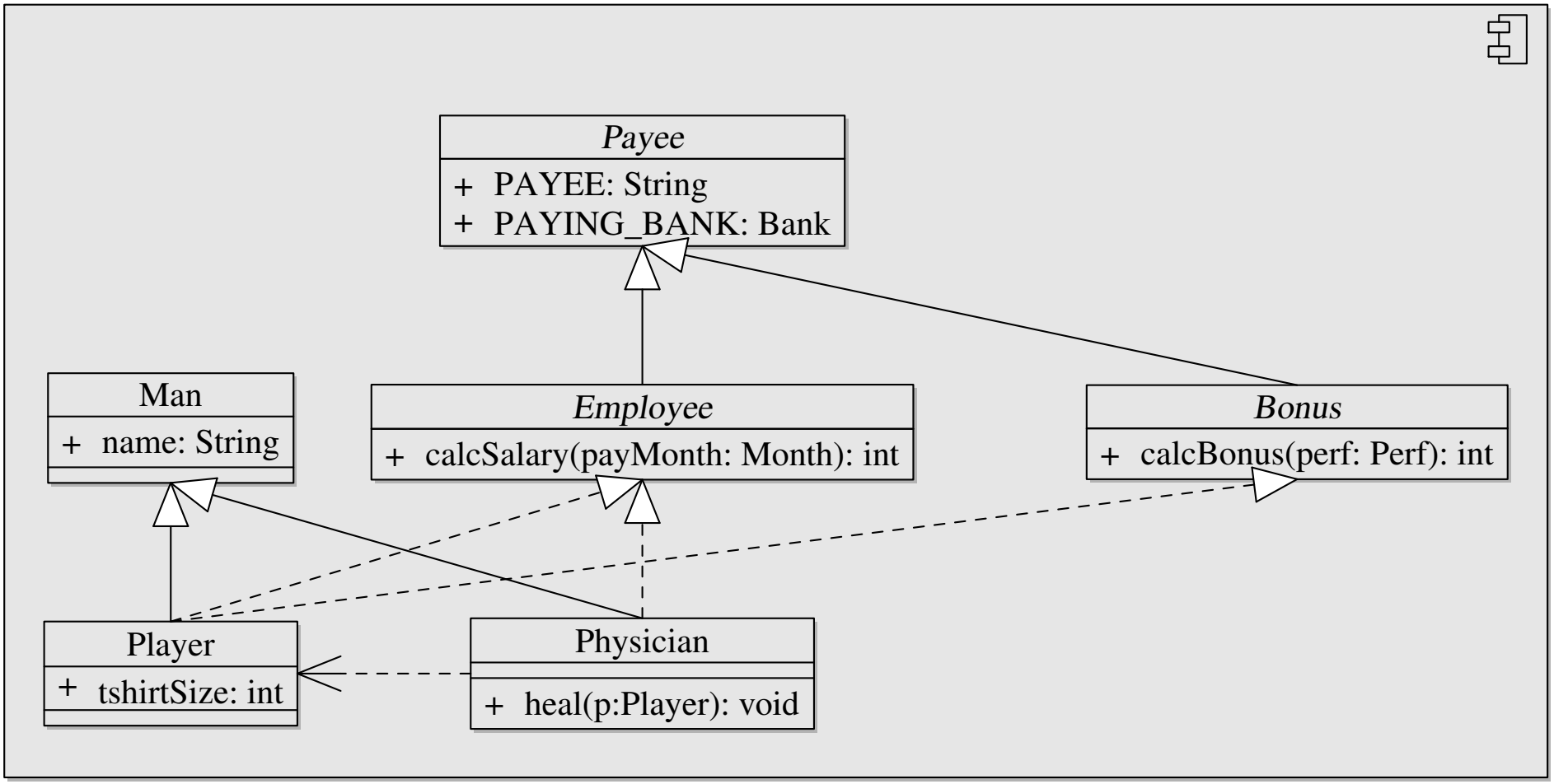
Curs 3: relatia dintre clase



Curs 4: incapsulare, static si final



Curs 5: clase abstracte si interfete



Clase interne (Nested Classes)

Nevoia de clase interne

Apar situatii in care:

- o clasa este folosita intr-un singur loc in program (in interiorul altei clase)
- o parte a continutului unei clase ar fi mai clar definita drept o alta clasa (dar are inca nevoie de acces la clasa parinte)

Ce este o clasa interna?

O clasa interna (Nested Class) se defineste in interiorul unei alte clase, denumita clasa externa (Outer Class),

Ce este o clasa interna?

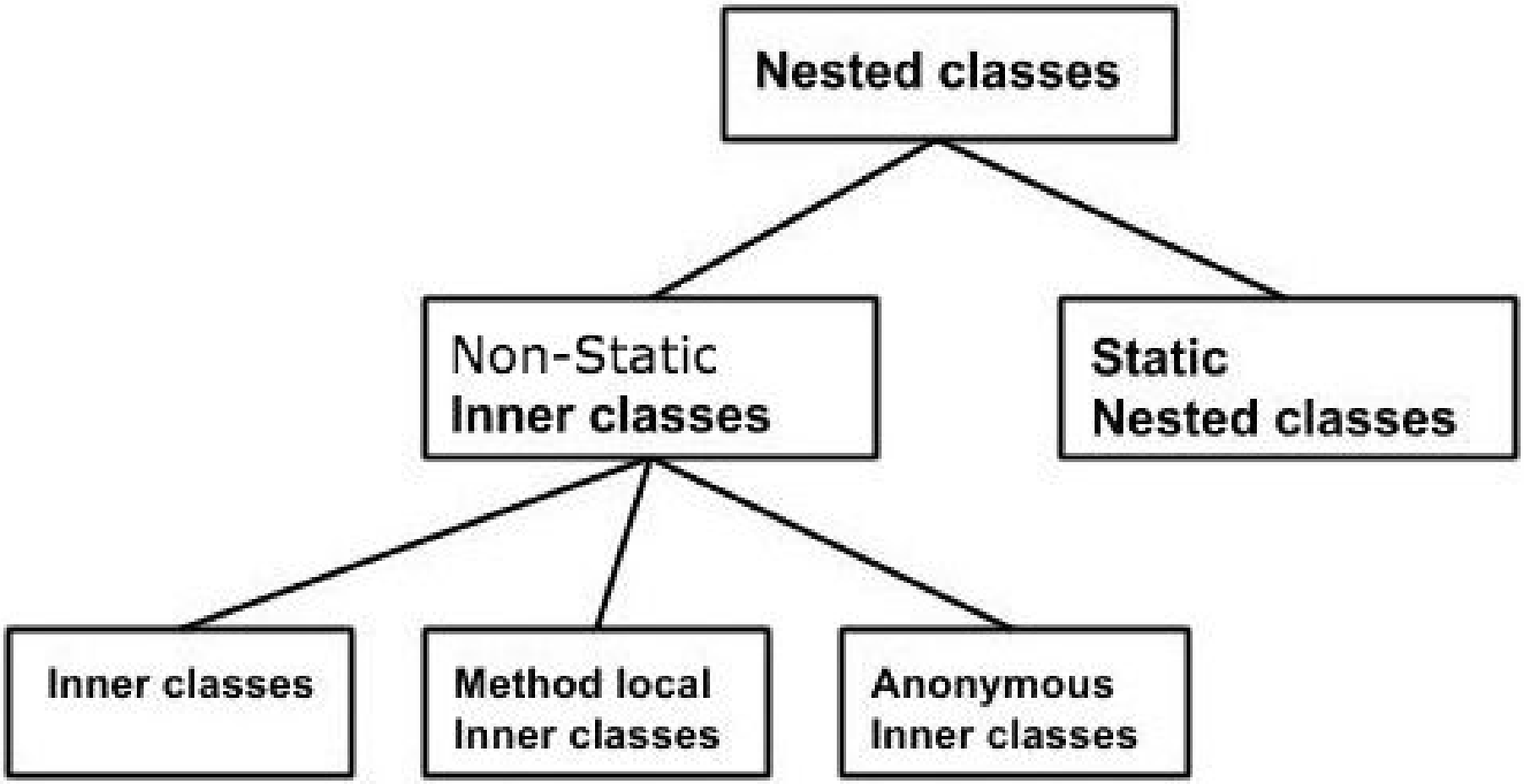
O clasa interna (Nested Class) se defineste in interiorul unei alte clase, denumita clasa externa (Outer Class), in diferite forme (ca membru, in interiorul unei metode, etc.).

Ce este o clasa interna?

O clasa interna (Nested Class) se defineste in interiorul unei alte clase, denumita clasa externa (Outer Class), in diferite forme (ca membru, in interiorul unei metode, etc.).

O clasa care nu este definita in interiorul unei alte clase se numeste Top-Level Class.

Tipuri de clase interne



▸ Inner Classes Tutorial

Inner Classes (clase interne normale)

Poate fi vizibila in exterior, caz in care poate fi accesata prin intermediul unei instante a clasei externe (ori ca rezultat al unei metode ori prin instantiere)

```
public class Person {
    public class Account {
        public String bank;
        public String iban;
    }
    public Account getAccount() {
        Account c = new Account("SwissBank", "CH23...");
        return c;
    }
}
public class Main {
    public static void main(String[] args) {
        Person pl = new Person();
        Person.Account c1 = pl.getAccount();
        Person.Account c2 = pl.new Account("USBank", "US23..
            ");
    }
```

Inner Classes (clase interne normale)

Putem sa accesam instanta clasei externe care a creat-o

```
public class Person {
    private name;
    public class Account {
        public String bank;
        public String iban;
        public String toString {
            return Person.this.name+"'s account: "+iban;
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Person pl = new Person("Mike");
        Person.Account c = pl.new Account("USBank", "US23.."
            );
        System.out.println(c); // ce se afiseaza?
    }
}
```

Inner Classes (clase interne normale)

Poate fi 'private', caz in care implementarea este ascunsa, dar interfata poate fi publica

```
interface BankAccount {
}
public class Person {
    private class Account implements BankAccount {
        public String bank;
        public String iban;
    }
    public BankAccount getAccount() {
        Account c = new Account("SwissBank", "CH23...");
        return c;
    }
}
public class Main {
    public static void main(String[] args) {
        Person p1 = new Person();
        BankAccount c3 = p1.getAccount(); // ok
    }
}
```

Method Local Inner Classes (clase interne in metode si blocuri)

Vizibilitatea clasei se limiteaza la metoda, respectiv bloc (desi codul binar se creeaza oricum). Pentru a putea folosi variabilele si parametrii metodei, clasa trebuie sa fie final.

```
public class TestMethodLocalInnerClass{
    public static void main(String[] args){
        class Greeter implements Runnable{
            private final String _greeted;
            public Greeter(String greeted){
                super();
                _greeted = greeted;
            }
            public void run(){
                System.out.printf("Hello %s!\n", _greeted);
            }
        }

        new Greeter("world").run();
        new Greeter("dog").run();
    }
```

Anonymous Inner Classes (clase anonime)

Daca o clasa este instantiata intr-un singur loc si apoi folosita (prin upcasting) doar prin intermediul clasei de baza sau al interfetei, numele sau nu mai este important.

```
new Thread( new Runnable() {
    Override
    public void run() {
        // do magic on separate thread
    }
}).start();
```

▸ More about Runnable

Anonymous Inner Classes (clase anonime)

Daca o clasa este instantiata intr-un singur loc si apoi folosita (prin upcasting) doar prin intermediul clasei de baza sau al interfetei, numele sau nu mai este important.

```
public class MyClass extends Applet {
    ...
        someObject.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                ...//Event listener implementation goes here
                    ...
            }
        });
    ...
    }
}
```

▸ More about Event Listeners

Anonymous Inner Classes (clase anonime)

Limitari:

- O clasa anonima poate sa extinda o singura clasa sau sa implementeze o singura interfata
- O clasa anonima nu are constructori (nu are nume, nu am sti cum se cheama constructorul); in mod implicit se invoca constructorul din clasa de baza care se potriveste cel mai bine cu parametrii folositi la initializare (sau constructorul default al clasei de baza daca nu sunt folositi parametrii)

Static Nested Classes (clase interne statice)

O clasa interna poate fi un membru static al clasei externe:

- nu va avea acces la instantele clasei externe (Outer.this), ci functioneaza ca o alta clasa top-level
- are sens sa fie interna pentru ca este folosita doar prin clasa externa (packaging convenience)
- nu avem nevoie de o instanta a clasei externe pentru a creea o instanta a clasei interne

```
LinkedList.Entry, Map.Entry
```

▸ How to use it

Exercitiu: Iterarea unui Map

Exercitiu: Iterarea unui Map

Hint: folositi Map.Entry

De citit

- Lab 6: Clase Interne
- More about Event Listeners