

Object - Oriented Programming

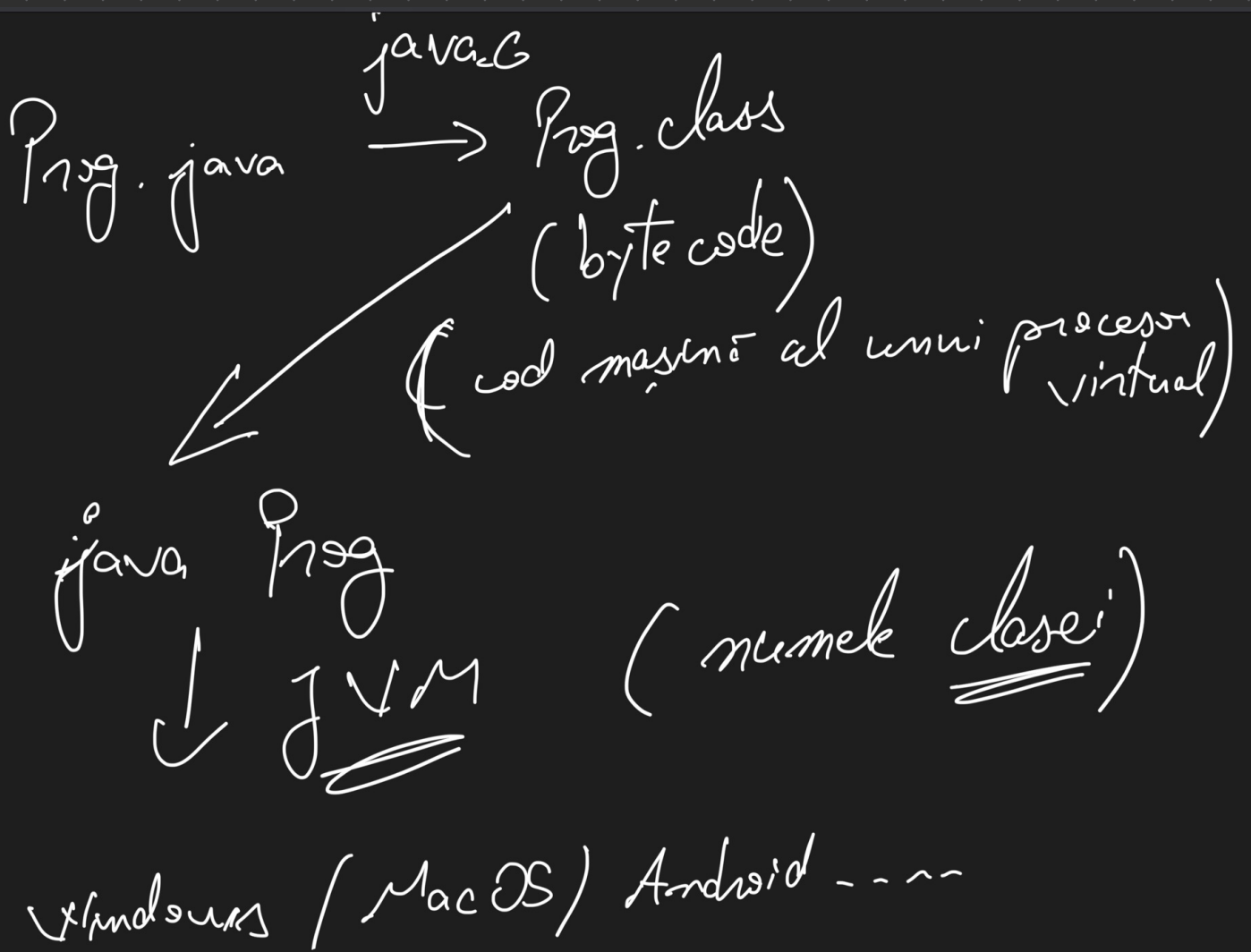
- Decompoziția în sub-sisteme (obiecte)

– sistemul este descompus într-un set de obiecte (și clasele lor)

```
class Main {
    public static void main(String[] args) {
        // ...
    }
}
```

Metoda main - aici începe programul

! orice clasă poate avea main-ul propriu



Portabilitate → "write once, run anywhere"

TIPLURI DE DATE PRIMITIVE

| | bits |
|---------|------|
| byte | 8 |
| short | 16 |
| int | 32 |
| long | 64 |
| float | 32 |
| double | 64 |
| char | 16 |
| boolean | - |

IDENTIFICATORI

key words

case - sensitive

Operatorii logici → se aplică doar
pe tip de date boolean

&& / ||
cu scurtcircuitate left → right

Operatorii relationali == !=
↓
return
value

True / False

< <=
> >=

Operatori pe biti:

\sim $|$ \ll \gg / \ggg $|&$ \wedge $|1$

Op. atribuire

$=$

$+=$

$*=$

$--$

Op. conditional

$exp1_bool \ ? \ exp1 \ : \ exp2$
 True False

CLASE ȘI OBIECTE

Clasa: def. un set de obiecte
care aceeași struct. și comp.

Obiect: instanță a unei clase

Clasa: implementarea unui obiect

```
class Clock {  
    private int hour, minute, second;  
    public void setTime(int h, int m, int s)  
}  
}
```

variabile referință

Clock clock1, clock2;

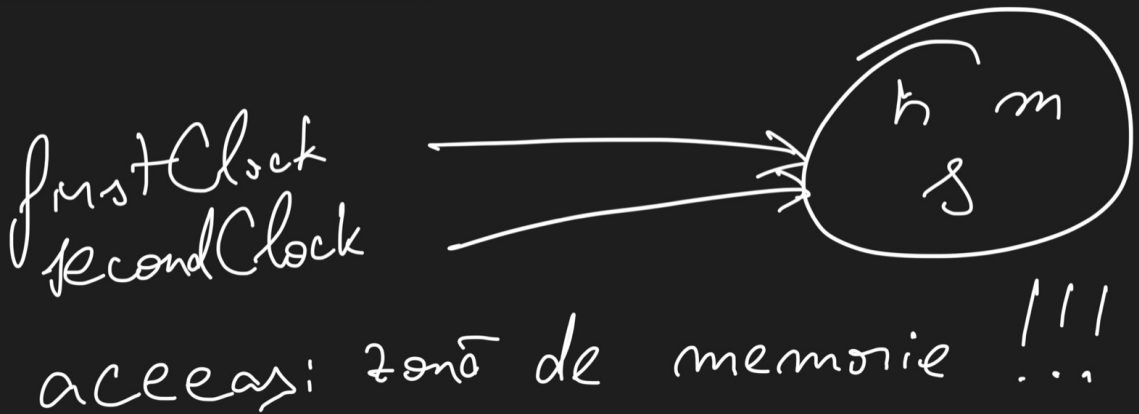
null: valoare ce arată
ca nu mai există
referință

clock1 = null;

Invocarea metodelor

clock } clock.setTime(5, 10, 0);
(int hour, ...)
this.hour = (hour >= 0) ...

ALIASING



Constructorii

- 1) Numele exact cu numele clasei
 - 2) Nu are tip returnat
- se poate executa o dată la crearea obiectului

```
class Clock {  
    public Clock(int h, m, s)  
    {  
    }  
}
```

Clock clock = new Clock(10, 10, 10);

- constructorul este generat automat
(fără argumente)
(no-arg)

Specificatori de acces

private → doar în interiorul clasei
public → accesat de oricine