

UN CASO DI STUDIO

Numeri Reali e Numeri Complessi

- Si vogliono progettare **due classi** che catturino il concetto di **numero reale** (`Real`) e di **numero complesso** (`Complex`)

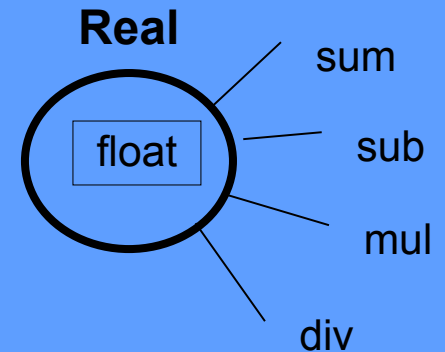
Principi-guida

- **aderenza alla realtà**
- **"efficienza" della rappresentazione**

Imposteremo un progetto per sottoporlo poi a *revisione critica*.

UN PRIMO APPROCCIO

- Un **progettista poco esperto** potrebbe iniziare subito a rappresentare il concetto di *numero reale*, senza riflettere a fondo.
- In un tale scenario, probabilmente per prima cosa verrebbe definita una classe **Real** simile a questa:
 - rappresentazione interna: un float
 - metodi per effettuare le quattro operazioni (sum, sub, mul, div)
 - ed eventualmente altre...



UN PRIMO APPROCCIO

Possibile realizzazione di Real:

```
class Real {  
    protected float val;  
  
    public Real(float x) { val = x; }  
  
    public Real sum(Real x) {  
        return new Real(val + x.val); }  
    public Real sub(Real x) {  
        return new Real(val - x.val); }  
    public Real mul(Real x) {  
        return new Real(val * x.val); }  
    public Real div(Real x) {  
        return new Real(val / x.val); }  
}
```

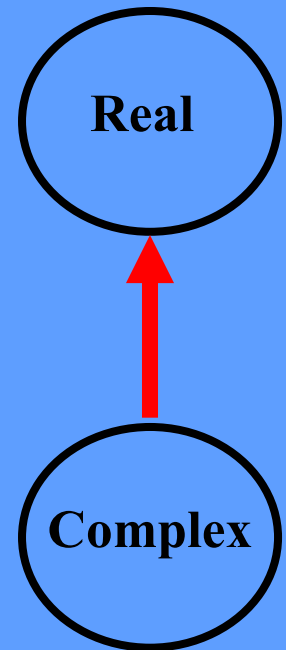
UN PRIMO APPROCCIO

Ora si deve introdurre la classe `Complex`.

- Anche qui, un **progettista poco esperto** potrebbe adottare, senza ben riflettere, un **approccio pragmatico**, e - poiché un `Complex` è caratterizzato da *due float* (parte reale e immaginaria), decidere di **derivare `Complex` da `Real`**
- È un approccio efficiente, perché riusa il float dei `Real` come parte reale, e aggiunge un nuovo float per la parte immaginaria...

MA la realtà è fatta a rovescio!

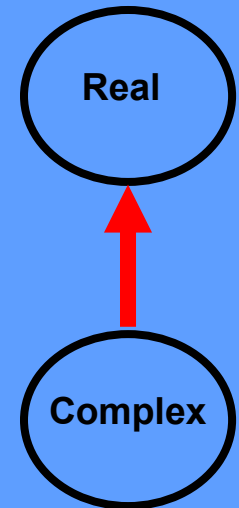
`Complex` un sottinsieme di `Real`??



UN PRIMO APPROCCIO

Sarebbe un **modello assurdo**

- si potrebbe assegnare un Complex a un Real, *ma non viceversa*
- le compatibilità di tipo funzionerebbero tutte “*a rovescio*” rispetto alla realtà matematica usuale.

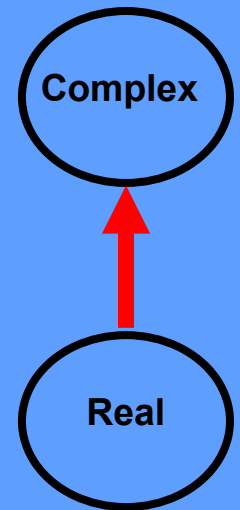


CONCLUSIONI

- è un **progetto sbagliato**, che non modella correttamente la realtà
- deve essere **ripensato da capo**.

UN NUOVO APPROCCIO

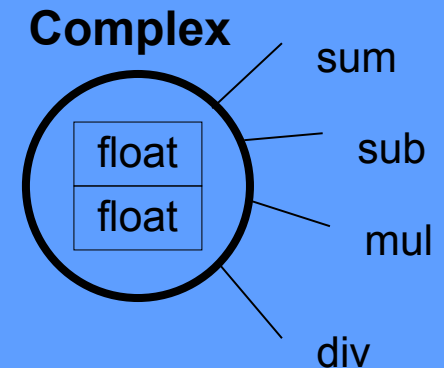
- Un **progettista esperto** riflette innanzitutto sulla **relazione esistente fra reali e complessi**
 - sono i reali ad essere un caso particolare dei complessi, non viceversa ($\mathbb{R} \subset \mathbb{C}$, ma $\mathbb{C} \not\subset \mathbb{R}$)
- Perciò, il **progettista esperto** decide *a priori* che la relazione corretta fra le due classi è che **Complex** sia la **classe-base**, e **Real** la classe derivata
 - l'efficienza viene *dopo*.



UN NUOVO APPROCCIO

- In questo nuovo scenario, **si definisce per prima la classe Complex**

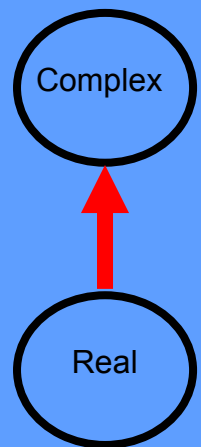
- rappresentazione interna: due float (*parte reale e parte immaginaria*)
- i metodi per effettuare le quattro operazioni (sum, sub, mul, div) devono essere in grado *di operare sui complessi*



- Poi, si deriva **Real** da essa

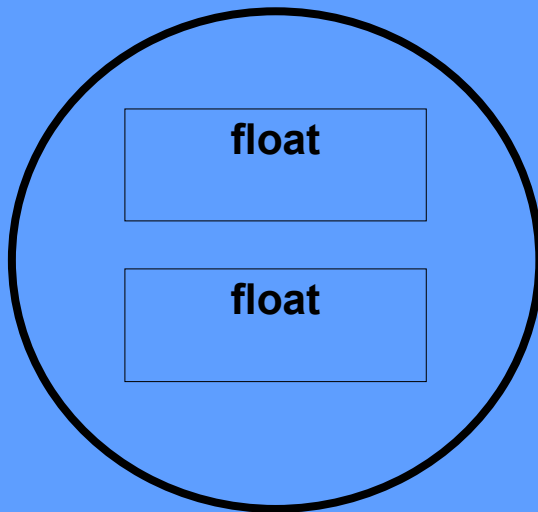
- è meno efficiente, perché i **Real** hanno comunque *due float*, anche se la parte immaginaria è sempre zero
- inoltre, tutte le operazioni sono calcolate fra **Complex**, in modo inutilmente complicato per i **Real**

- **Ma la realtà è modellata correttamente**



NUOVO APPROCCIO: Complex

Progetto della classe Complex



OPERAZIONI

- somma fra complessi
- sottrazione fra complessi
- moltiplicazione fra complessi
- divisione di un complesso per un fattore reale
- coniugato di un numero complesso
- modulo (al quadrato) di un complesso
- divisione fra complessi

RELAZIONI UTILI

- $(a+ib) \pm (c+id) = (a+c) \pm i (b+d)$
- $(a+ib) / x = (a/x + i b/x) \quad x \in R$
- $z / w = (z \times \text{cgt}(w)) / |w|^2$

RELAZIONI UTILI

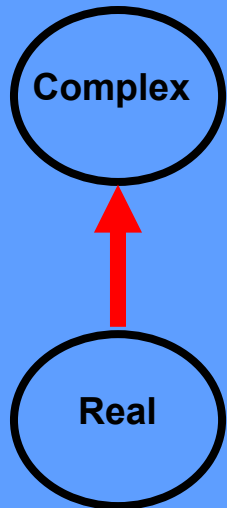
- $(a+ib) \times (c+id) = (ac-bd) + i (bc + ad)$
- $\text{cgt}(a+ib) = (a-ib)$

NUOVO APPROCCIO: Complex

```
public class Complex {
    protected float re, im;
    public Complex(float r, float i) { re = r; im = i; }
    public Complex sum(Complex z){
        return new Complex(re+z.re, im+z.im); }
    public Complex sub(Complex z){
        return new Complex(re-z.re, im-z.im); }
    public Complex mul(Complex z){
        return new Complex(re*z.re-im*z.im, im*z.re+re*z.im); }
    public Complex div(Complex z){
        return mul(cgt(z)).divByFactor(z.squaredModule()); }
    public Complex cgt(Complex z){ return new Complex(re,-im); }
    public Complex divByFactor(float x) {
        return new Complex(re/x, im/x); }
    public float squaredModule(){ return re*re + im*im; }
    public String toString(){ return "" + re + "+i" + im;}
}
```

NUOVO APPROCCIO: Real

Progetto della classe Real



- Ogni numero reale ha comunque una parte immaginaria, che però vale zero
 - Le operazioni precedenti restano valide, ma risultano perciò *inutilmente inefficienti*
 - Può essere utile re-implementarle per guadagnare efficienza
-
- Infatti, somma / sottrazione / moltiplicazione /divisione fra reali sono *molto più semplici* delle corrispondenti operazioni fra complessi...
 - .. e il risultato è un reale

NUOVO APPROCCIO: Real

```
public class Real extends Complex {  
    public Real(float x) { super(x,0); }  
    public Real sum(Real x){ return new Real(re + x.re); }  
    public Real sub(Real x){ return new Real(re - x.re); }  
    public Real mul(Real x){ return new Real(re * x.re); }  
    public Real div(Real x){ return new Real(re / x.re); }  
    public String toString(){ return "" + re;}  
}
```

NOTARE:

- il costruttore di Real si appoggia su quello di Complex, impostando però sempre la parte immaginaria a zero
- le quattro operazioni sono reimplementate *nel caso specifico di entrambi operandi Real*, nel qual caso il risultato è a sua volta un Real
- rimangono comunque disponibili tutte le operazioni già esistenti
- viene reimplementata la toString() in modo specifico ai Real

ESEMPIO D'USO

```
public class Prova {  
    public static void main(String args[]){  
        Real r1 = new Real(18.5F), r2 = new Real(3.14F);  
        Complex c1 = new Complex(-16, 0), c2 = new Complex(3, 2),  
            c3 = new Complex(0, -2);  
        Real r = r1.sum(r2); Complex c = c1.sum(c2);  
        System.out.println("r1 + r2 = " + r); // il reale 21.64  
        System.out.println("c1 + c2 = " + c); // il complex -13+2i  
        System.out.println("c1 + c2 -i = " + c.sub(new Complex(0,1)));  
                                                    // il complesso -13+i  
        c = c.sum(c3);  
        System.out.println("c + c3 = " + c); // -13+0i  
        c = r;  
        System.out.println("c = r; c = " + c); // qui c è reale  
        // POLIMORFISMO: scatta la toString dei reali, stampa 21.64  
    }  
}
```

```
r1 + r2 = 21.64  
c1 + c2 = -13.0+i2.0  
c1 + c2 -i = -13.0+i1.0  
c + c3 = -13.0+i0.0  
c = r; c = 21.64
```