## 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`)**

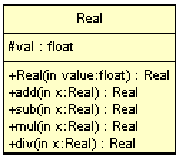
**Principi-guida**

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

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

## UN PRIMO APPROCCIO

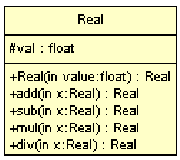
- **Un progettista *poco esperto* potrebbe iniziare subito a rappresentare il concetto di *numero reale*, senza riflettere a fondo.**
- **In un tale scenario, probabilmente verrebbe progettata una classe `Real` come questa:**
  - rappresentazione interna: un float
  - costruzione a partire da un valore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




## PRIMO APPROCCIO: Real

**Possibile realizzazione di questo `Real`:**

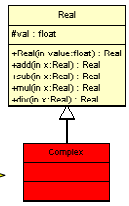
```
class Real {
 protected float val;
 public Real(float value) { val = value; }
 public Real sum(Real x){
   return new Real(val + x.val); }
 public Real sub(Real x){
   return new Real(val – x.val); }
 public Real mul(Real x){
   return new Real(val * x.val); }
 public Real div(Real x){
   return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




## PRIMO APPROCCIO: Complex

**Ora si deve definire la classe `Complex`.**

- **un progettista *poco esperto*, considerando che un `Complex` è caratterizzato da parte reale e immaginaria, potrebbe decidere di *derivare `Complex` da `Real`.***
- ***MA la realtà è fatta a rovescio! I complessi NON SONO un sottoinsieme dei reali!***
- **NON modella correttamente la realtà**
- **È errato, deve essere ripensato da capo!**

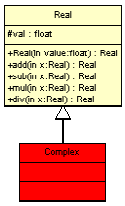



**Modello falso della realtà !**

## PRIMO APPROCCIO: bilancio

**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.**




**Dove sta l'errore di fondo?**

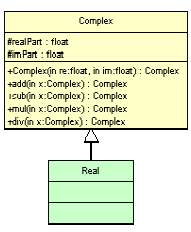
- **La parte immaginaria *non è una proprietà extra dei numeri complessi: ce l'hanno anche i reali*, tanto è vero che *sappiamo perfino quanto vale!* (zero)**
- **Nei reali non si indica solo perché è sottintesa, *non perché non ci sia!* L'abitudine tende a fuorviare…!**

## UN NUOVO APPROCCIO

- **Un progettista *esperto* riflette innanzitutto *sulla relazione esistente fra reali e complessi***

**$R \subset C$**

- **Perciò, egli decide *a priori* che il modello corretto è che `Complex` generalizzi `Real`**
  - le considerazioni di efficienza, se mai, vengono *dopo*




**Modello corretto della realtà**

## NUOVO APPROCCIO: caratteristiche

### La classe `Complex`

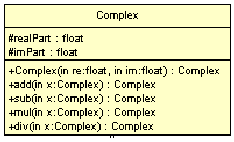
- **IPOTESI: rappresentazione interna basata su due float**
  - *parte reale, parte immaginaria*
- **REQUISITO: le operazioni** (sum, sub, mul, div) **devono poter operare *su qualunque numero complesso***

### La classe derivata `Real`

- **ogni istanza ha comunque *due float*, anche se la parte immaginaria è sempre zero *(inefficienza)***
- **le operazioni continuano a operare sui complessi, sebbene possano essere semplificate.**

## LA CLASSE `Complex`

### Progetto della classe `Complex`




**OPERAZIONI**
- somma fra complessi
- sottrazione fra complessi
- moltiplicazione fra complessi
- divisione fra complessi

*... e inoltre*

- divisione di un complesso per un fattore reale
- coniugato di un numero complesso
- modulo (al quadrato) di un complesso

**RELAZIONI UTILI**
- $(a+ib) \pm (c+id) = (a+c) \pm i\,(b+d)$
- $(a+ib) / x = (a/x + i\,b/x) \quad x \in R$
- $z / w = (z \times cgt(w)) / |w|^2$

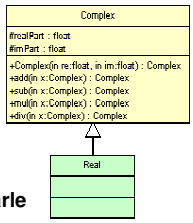
**RELAZIONI UTILI**
- $(a+ib) \times (c+id) = (ac-bd) + i\,(bc + ad)$
- $cgt(a+ib) = (a-ib)$

## `Complex`: IMPLEMENTAZIONE

```
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;}
}
```

## LA CLASSE `Real`

- **Ogni numero reale ha comunque una parte immaginaria, che però vale zero**
- **Di conseguenza, anche se le operazioni precedenti restano valide, risultano *inutilmente inefficienti***
- **Ergo, può essere utile re-implementarle per guadagnare efficienza**
  - **somma, sottrazione, moltiplicazione e divisione fra reali sono *molto più semplici* delle stesse operazioni fra complessi**
  - **il loro risultato, inoltre, è sempre un reale**




## `Real`: IMPLEMENTAZIONE

```
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 operazioni sono reimplementate *nel caso specifico in cui gli operandi siano entrambi Real*, poiché allora anche il risultato lo è**
- **rimangono comunque disponibili *tutte* le operazioni già esistenti**
- **viene reimplementata `toString()` in modo specifico per i 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);
  // POLIMORFISMO: c è reale → 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
```