Sesta esercitazione
Linguaggi di programmazione

Tutor didattico: Giosuè Cotugno
giosue.cotugno2@unibo.it
A.A. 2023/2024

Esercizio 1

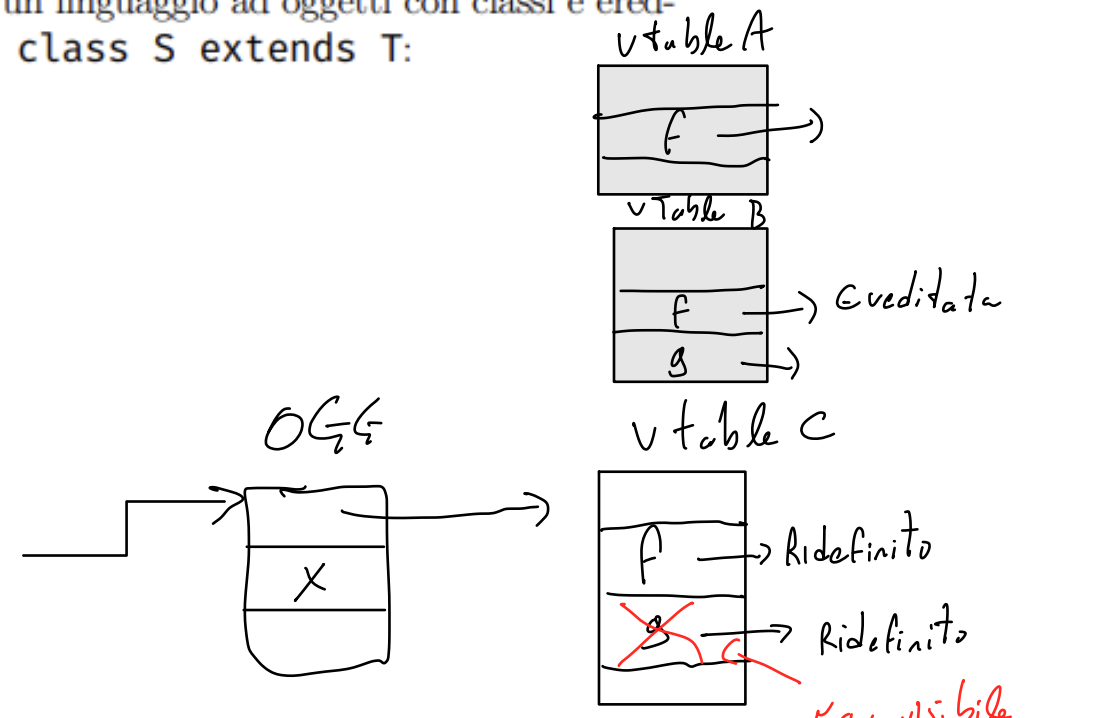
In un linguaggio ad oggetti con selezione dinamica dei metodi, si dica cosa stampa il seguente frammento:

```
class A {
 int x = 4;
 void s3(){
  x = 3;
 }
 void f(){
  x = 8;
 }
}
class B extends A {
 int x = 5;
 void f (){
  x = 10;
  s3();
 }
}
A a = new B();
B b = ( B ) a;
a.f();
print( b.x );
print( a.x );
```

Esercizio 2

Sono date le seguenti dichiarazioni in un linguaggio ad oggetti con classi e ereditarietà singola, espressa dalla sintassi `class S extends T`:

```
class A {
 int x;
 void f(){}
}
class B extends A {
 void g (){}
}
class C extends B {
 int x;
 void f(){}
 void g(){}
}
A a = new C (); // OGG
```

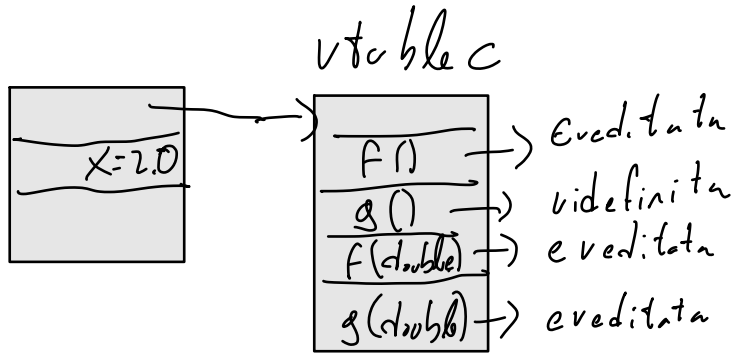


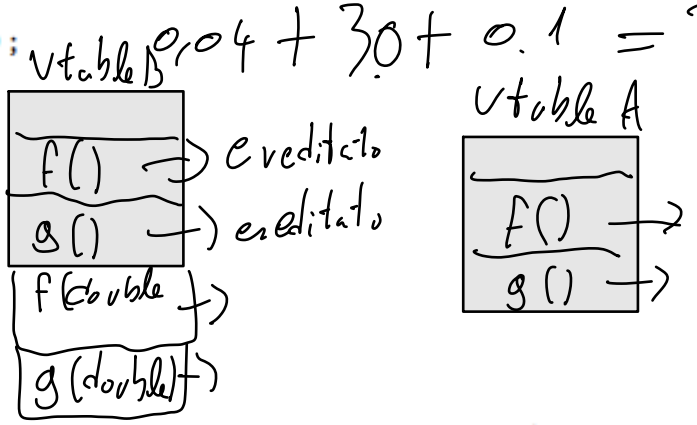
Si dia una rappresentazione grafica, con breve descrizione, dell'oggetto `OGG` e le vtable delle tre classi in memoria.

Esercizio 3

8. Si consideri il seguente frammento di codice Java

```
class A {
 double x = 3.0;
 double f(){ return x; }
 double g(){ return f(); }
}
class B extends A {
 double x = 2.0;
 double f( double x ){ return x / 10; }
 double g( double x ){
  return f( f( g() + f() ) ) + f() + f( g() );
 }
}
class C extends B {
 double g(){ return f() - x; }
}

B c = new C();
write( c.g( c.x ) );
```





Indicare l'output della funzione `write`, che accetta un `double` e lo stampa a schermo nel formato `parte_intera.parte_decimale`. Spiegare brevemente il ragionamento seguito e dare una possibile rappresentazione delle vtable corrispondenti alle classi.

Esercizio 4

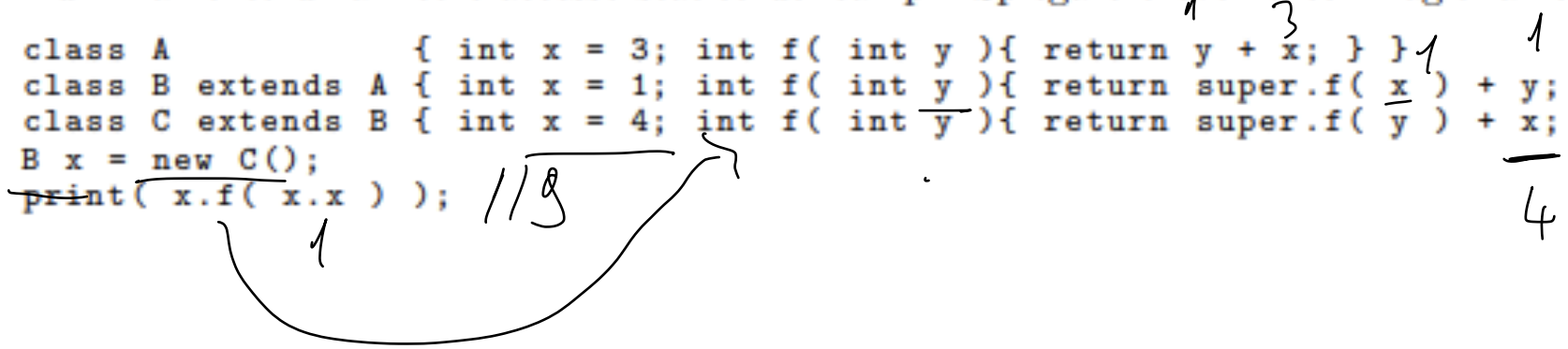
8. Cosa stampa il seguente frammento di codice Java?

```
class A {
 float a = 0;
 float a(){ return ++a; }
 float f(){
  if( ( a = a() + a ) < 27 ){
   System.out.println( a );
   return f();
  }
  return a;
 }
}

class B extends A {
 float a = 1;
 float a(){ return a++; }
}

new B().f();
```

Esercizio 5

8. Cosa stampa il frammento di codice sotto, considerando di avere un linguaggio ad oggetti che supporta ereditarietà (`B extends A` indica che `B` eredita da `A` e `super` riferisce l'oggetto come super-classe), indirizzamento dinamico e accesso statico dei campi? Spiegare brevemente il ragionamento seguito.

```
class A             { int x = 3; int f( int y ){ return y + x; } }
class B extends A { int x = 1; int f( int y ){ return super.f( x ) + y; } }
class C extends B { int x = 4; int f( int y ){ return super.f( y ) + x; } }
B x = new C();
print( x.f( x.x ) );
```



Esercizio 6

Sono date, in Java, le seguenti dichiarazioni di classi:

```
class A {
 int a = 10;
  void f (){ g(); }
  void g (){ a = 5; }
}
class B extends A {
  int a = 7;
  int b = 2;
  void g (){ b = 15; }
}
```

Si dica cosa stampa, nello scope di queste dichiarazioni, il seguente frammento (la stampa usa il metodo statico `System.out.println`, che stampa l'input dato e una nuova linea):

```
A y = new A();
B x = new B();
x.f();
y = x;
System.out.println( x.a );
System.out.println( y.a );
System.out.println( x.b );
```

Esercizio 7

Lion <: Carnivore <: Mammal <: Animal

giraffe <: Herbivore <: Mammal

T[? <: S] Covarianza

T[? :> S] Controvarianza

5. In un linguaggio con passaggio per riferimento e supporto al polimorfismo di sottotipo e parametrico, sono dati i seguenti tipi per cui vale la relazione di sottotipaggio `<:` nelle seguenti direzioni: `Mammal <: Animal`, `Lion <: Carnivore <: Mammal` e `Giraffe <: Herbivore <: Mammal`. Viene inoltre definito il contenitore polimorfo `Cage[ T ]` dotato delle operazioni `add: T -> ()` e `remove: () -> T`, con `T` parametro di tipo. Il linguaggio offre l'istruzione `new` per creare una nuova istanza di un tipo e supporta sottotipi parametrici con la notazione `T[? <: S]` e `T[? :> S]` per indicare la relazione di sottotipaggio del tipo parametrico `?` rispetto ad un tipo concreto `S`.

   Nel codice sottostante, indicare quali istruzioni sono errate e spiegare brevemente perchè.

```
Cage[ ? :> Carnivore ] cage1 = new Cage[ Mammal ]();    // I1
Cage[ ? <: Mammal ]    cage2 = new Cage[ Carnivore ](); // I2
Cage[ ? <: Herbivore ] cage3 = new Cage[ Giraffe ]();   // I3
cage1.add( new Lion() );                                // I4
Mammal a1 = cage2.remove();                             // I5
cage1.add( new Giraffe() );                             // I6
cage2.add( cage1.remove() );                            // I7
cage2.add( a1 );                                        // I8
Herbivore a2 = cage3.remove();                          // I9
cage2 = cage3;                                          // I10
cage3.add( a2 );                                        // I11
cage1.add( cage3.remove() );                            // I12
```

Carnivore <: Mammal

Cage[Carnivore] :> Cage[Mammal]