1

Protezione
Oggetti in stack
Ripasso del polimorfismo

2

Protezione

Access from within class's package

Access Modifier	Inherited	Accessible
default (no modifier)	Yes	Yes
Public	Yes	Yes
Protected	Yes	Yes
Private	No	No

Access outside of a package T

Access Modifier	Inherited	Accessible
default (no modifier)	No	No
Public	Yes	Yes
Protected	Yes	No
Private	No	No

3

```
package pila;
import ext.*;
class A {
   public static void main(String a[]) {
     B b=new B();
     // b.a=3; ACCESSO VIETATO!
     // b.f(); ACCESSO VIETATO
     b.g();
   }
};
```


```
package ext;
public class B extends C{
  public void g() {
    a=3;
    System.out.println("g");
    System.out.println(a);
    f();
  }
}
class C  {
   protected int a=1;
   protected void
    f(){System.out.println("f");}
}
```

4

```
package pila;

class A {
   public static void main(String a[]) {
     B b=new B();
     b.a=3; ACCESSO PERMESSO!
     b.f(); ACCESSO PERMESSO!
     b.g();
   }
};
```


```
package pila;
public class B extends C{
  public void g() {
    a=3;
    System.out.println("g");
    System.out.println(a);
    f();
  }
}
class C  {
   protected int a=1;
   protected void
    f(){System.out.println("f");}
}
```

5

```
#include <iostream.h>
int main () {
   B *b=new B;
   //b->a=3; ACCESSO VIETATO!
   //b->f(); ACCESSO VIETATO!
   b->g();
}
```

```
class A {
protected:
   int a;
   void f(){cout<<"f"<<endl;}
} ;
class B:public A {
   public:
   void g() {
     a=3;
     cout<<"g"<<endl;
     cout<<a<<endl;
     f();
   }
} ;
```

A * o=new A;
o.a=3; //vietato

6

```
#include <iostream.h>
int main () {
   B *b=new B;
   //b->a=3; ACCESSO VIETATO!
   //b->f(); ACCESSO VIETATO!
   b->g();
}
```

```
class A {
public:
   int a;
   void f(){cout<<"f"<<endl;}
} ;
class B:private A {
   public:
   void g() {
     a=3;
     cout<<"g"<<endl;
     cout<<a<<endl;
     f();
   }
} ;
```

la combinazione di accesso pubblico e ereditarietà privata dà risultati simili al protected

A * o=new A;
o.a=3; //permesso

Oggetti in Stack

```
int main() {
   //Una Pila in Stack…
   Pila s; // oppure Pila s(5);

   for (int k=1; k<10;k++) s.inserisci(k);
   cout<<"s";s.stampaStato();
   for (int k=1; k<8;k++)
      cout<<s.estrai()<<endl;

   Pila *w=&s;
   //NO! delete w;
}
```

Re-implementazione del main

9

Distruttori

```
Pila()
```

```
int main() {
  Pila *c=new Pila;
  return 0;
}
```

Memory leak!

```
Pila()
~Pila()
```

```
int main() {
  Pila c;
  return 0;
}
```

10

Ripasso:
Esempio con i fischietti

11

```
class Fischietto {
  public:
      void fischia();
      void fischia(int n);
      void cadi();
};
```

```
#include "Fischietto.h"
      void Fischietto::fischia() {
         cout<<"Fiii ";
         cout<<endl;
      }
      void Fischietto::fischia(int n){
        for (int k=0; k<n; k++)
          cout<<"Fiii ";
        cout<<endl;
      }
      void Fischietto::cadi() {
         cout<<"Toc ";
         cout<<endl;
      }
```

12

```
class FischiettoBitonale:public Fischietto{
  public:
      void fischia();
      void fischia(int n);
  };
```

Ereditarietà (Inheritance)

```
#include "FischiettoBitonale.h"
      void FischiettoBitonale::fischia() {
          cout<<"FuuuFiii ";
          cout<<endl;
      }
      void FischiettoBitonale::fischia(int n){
        for (int k=0; k<n; k++)
          cout<<"FuuuFiii ";
        cout<<endl;
      }
```

Method overriding

13

```
int main()
{
  Fischietto f;
  f.fischia();
  f.fischia(3);
  f.cadi();

  FischiettoBitonale fb;
  fb.fischia();
  fb.fischia(3);
  fb.cadi();
```

```
int tipo=0;
do {
  cout<<"Scegli [1:normale - 2:bitonale] :";
  cin>>tipo;
} while (tipo<1 || tipo>2);
Fischietto * x;
switch (tipo) {
  case 1: x=&f; break;
  case 2: x=&fb; break;
  default:
     cout<<"qui non deve mai passare!";
}
x->fischia();
x->fischia(4);
x->cadi();
return 0;
}
```

14

```
Fiii
Fiii Fiii Fiii
Toc

FuuuFiii
FuuuFiii FuuuFiii FuuuFiii
Toc

Scegli [1:normale - 2:bitonale] :2
Fiii
Fiii Fiii Fiii Fiii
Toc
```

STATIC BINDING

15

```
class Fischietto {
  public:
      virtual void fischia();
      void fischia(int n);
      void cadi();
};
```

```
#include "Fischietto.h"
      void Fischietto::fischia() {
         cout<<"Fiii ";
         cout<<endl;
      }
      void Fischietto::fischia(int n){
        for (int k=0; k<n; k++)
          cout<<"Fiii ";
        cout<<endl;
      }
      void Fischietto::cadi() {
         cout<<"Toc ";
         cout<<endl;
      }
```

16

```
Fiii
Fiii Fiii Fiii
Toc

FuuuFiii
FuuuFiii FuuuFiii FuuuFiii
Toc

Scegli [1:normale - 2:bitonale] :2
FuuuFiii  ←DYNAMIC BINDING
Fiii Fiii Fiii Fiii  ← STATIC BINDING
Toc
```

17

```
class Fischietto {
  public:
      virtual void fischia();
      virtual void fischia(int n);
      void cadi();
};
```

```
#include "Fischietto.h"
      void Fischietto::fischia() {
         cout<<"Fiii ";
         cout<<endl;
      }
      void Fischietto::fischia(int n){
        for (int k=0; k<n; k++)
          cout<<"Fiii ";
        cout<<endl;
      }
       void Fischietto::cadi() {
         cout<<"Toc ";
         cout<<endl;
      }
```

18

```
Fiii
Fiii Fiii Fiii
Toc

FuuuFiii
FuuuFiii FuuuFiii FuuuFiii
Toc

Scegli [1:normale - 2:bitonale] :2
FuuuFiii
FuuuFiii FuuuFiii FuuuFiii FuuuFiii
Toc
```

DYNAMIC BINDING

19

```
int main()
{
  Fischietto f;
  f.fischia();
  f.fischia(3);
  f.cadi();

  FischiettoBitonale fb;
  fb.fischia();
  fb.fischia(3);
  fb.cadi();
```

```
int tipo=0;
do {
  cout<<"Scegli [1:normale - 2:bitonale] :";
  cin>>tipo;
} while (tipo<1 || tipo>2);
Fischietto x;
switch (tipo) {
 case 1: x=f; break;
 case 2: x=fb; break;
 default:
    cout<<"qui non deve mai passare!";
}
x.fischia();
x.fischia(4);
x.cadi();
return 0;
}
```

20

```
Fiii
Fiii Fiii Fiii
Toc

FuuuFiii
FuuuFiii FuuuFiii FuuuFiii
Toc

Scegli [1:normale - 2:bitonale] :2
Fiii
Fiii Fiii Fiii Fiii
Toc
```

Il dynamic binding NON SI APPLICA alle variabili (solo a puntatori)

21

```
int main()
{
 Fischietto f;
 FischiettoBitonale f;
 int tipo=0;
   do {
      cout<<"Scegli [1:normale - 2:bitonale] :";
      cin>>tipo;
   } while (tipo<1 || tipo>2);
   switch (tipo) {
     case 1: esegui(&f); break;
     case 2: esegui(&fb); break;
     default: cout<<"qui non deve mai passare!";
   }
   int h;cin>>h;
   return 0;
}
```

```
void esegui(Fischietto * x) {
   x->fischia();
   x->fischia(4);
   x->cadi();
}
```

```
Scegli [1:normale - 2:bitonale] :2
FuuuFiii
FuuuFiii FuuuFiii FuuuFiii FuuuFiii
Toc
```

22

```
int main()
{
 Fischietto f;
 FischiettoBitonale f;
 int tipo=0;
   do {
      cout<<"Scegli [1:normale - 2:bitonale] :";
      cin>>tipo;
   } while (tipo<1 || tipo>2);
   switch (tipo) {
     case 1: esegui(f); break;
     case 2: esegui(fb); break;
     default: cout<<"qui non deve mai passare!";
   }
   int h;cin>>h;
   return 0;
}
```

```
void esegui(Fischietto & x) {
   x.fischia();
   x.fischia(4);
   x.cadi();
}
```

```
Scegli [1:normale - 2:bitonale] :2
FuuuFiii
FuuuFiii FuuuFiii FuuuFiii FuuuFiii
Toc
```

23

```
int main()
{
 Fischietto f;
 FischiettoBitonale f;
 int tipo=0;
   do {
      cout<<"Scegli [1:normale - 2:bitonale] :";
      cin>>tipo;
   } while (tipo<1 || tipo>2);
   switch (tipo) {
     case 1: esegui(f); break;
     case 2: esegui(fb); break;
     default: cout<<"qui non deve mai passare!";
   }
   return 0;
}
```

```
void esegui(Fischietto x) {
   x.fischia();
   x.fischia(4);
   x.cadi();
}
```

```
Scegli [1:normale - 2:bitonale] :2
Fiii
Fiii Fiii Fiii Fiii
Toc
```

24

Costruttori e distruttori

```
#include <iostream.h>
class Fischietto {
 public:
 Fischietto() { cout<<"creating Fischietto "<<endl; }
 ~ Fischietto () { cout<<"destroying Fischietto "<<endl; }
};

class FischiettoBitonale:public Fischietto {
 public:
 FischiettoBitonale() { cout<<"creating FischiettoBitonale "<<endl; }
  ~ FischiettoBitonale() { cout<<"destroying FischiettoBitonale "<<endl;}
};

int main()
{
       FischiettoBitonale p;
       return 0;
}
```

```
OUTPUT:
creating Fischietto
creating FischiettoBitonale
destroying FischiettoBitonale
destroying Fischietto
```

25

Costruttori e distruttori

```
#include <iostream.h>
class Fischietto {
 public:
 Fischietto() { cout<<"creating Fischietto "<<endl; }
 ~ Fischietto () { cout<<"destroying Fischietto "<<endl; }
};

class FischiettoBitonale:public Fischietto {
 public:
 FischiettoBitonale() { cout<<"creating FischiettoBitonale "<<endl; }
  ~ FischiettoBitonale() { cout<<"destroying FischiettoBitonale "<<endl;}
};

int main()
{
  FischiettoBitonale *p = new FischiettoBitonale ;
  delete p;
  return 0;
}
```

OUTPUT:
creating Fischietto
creating FischiettoBitonale
destroying FischiettoBitonale
destroying Fischietto

26

Costruttori e distruttori

```
#include <iostream.h>
class Fischietto {
 public:
 Fischietto() { cout<<"creating Fischietto "<<endl; }
 ~ Fischietto () { cout<<"destroying Fischietto "<<endl; }
};

class FischiettoBitonale:public Fischietto {
 public:
 FischiettoBitonale() { cout<<"creating FischiettoBitonale "<<endl; }
  ~ FischiettoBitonale() { cout<<"destroying FischiettoBitonale "<<endl;}
};

int main()
{
  Fischietto *p = new FischiettoBitonale ;
  delete p;
  return 0;
}
```

OUTPUT:
creating Fischietto
creating FischiettoBitonale
destroying Fischietto

27

Costruttori e distruttori

```
#include <iostream.h>
class Fischietto {
 public:
 Fischietto() { cout<<"creating Fischietto "<<endl; }
 virtual ~ Fischietto () { cout<<"destroying Fischietto "<
};

class FischiettoBitonale:public Fischietto {
 public:
 FischiettoBitonale() { cout<<"creating FischiettoBitonale "<<endl; }
  virtual ~ FischiettoBitonale()
                { cout<<"destroying FischiettoBitonale "<<endl;}
};

int main()
{
  Fischietto *p = new FischiettoBitonale ;
 delete p;
  return 0;
}
```

OUTPUT:
creating Fischietto
creating FischiettoBitonale
destroying FischiettoBitonale
destroying Fischietto