# Esercizi di Programmazione ad Oggetti

## Lista n. 4

### Esercizio 1

Definire una superclasse `ContoBancario` e due sue sottoclassi `ContoCorrente` e `ContoDiRisparmio` che soddisfano le seguenti specifiche:

- Ogni `ContoBancario` è caratterizzato da un saldo e rende disponibili due funzionalità di deposito e prelievo: `int deposita(int)` e `int preleva(int)` che ritornano il saldo aggiornato dopo l'operazione di deposito/prelievo.
- Ogni `ContoCorrente` è caratterizzato anche da una spesa fissa uguale per ogni `ContoCorrente` che deve essere detratta dal saldo ad ogni operazione di deposito e prelievo.
- Ogni `ContoDiRisparmio` deve avere un saldo non negativo e pertanto non tutti i prelievi sono permessi; d'altra parte, le operazioni di deposito e prelievo non comportano costi aggiuntivi e restituiscono il saldo aggiornato.
- Si definisca inoltre una classe `ContoArancio` derivata da `ContoDiRisparmio`. La classe `ContoArancio` deve avere un `ContoCorrente` di appoggio: quando si deposita una somma $S$ su un `ContoArancio`, $S$ viene prelevata dal `ContoCorrente` di appoggio; d'altra parte, i prelievi di una somma $S$ da un `ContoArancio` vengono depositati nel `ContoCorrente` di appoggio.

**Soluzione.**

```
//SOLUZIONE POSSIBILE

#include<iostream>
using std::cout; using std::endl;

class ContoBancario {
protected:
    int saldo;
public:
    ContoBancario(int x=0): saldo(x) {if (saldo <0) saldo=0;}
    int Saldo() const {return saldo;}
    int deposita(int d) {return saldo +=d;}
    int preleva(int p) {return saldo -=p;}
};


class ContoCorrente: public ContoBancario {
private:
    static int spesaFissa;
public:
    int deposita(int d) { return saldo = saldo + d - spesaFissa; }
    int preleva(int p) { return saldo = saldo - p - spesaFissa; }
};
int ContoCorrente::spesaFissa = 1;

class ContoDiRisparmio: public ContoBancario {
  public:
  int preleva(int p) {
    if (p > saldo) {
      cout << "Il prelievo non e' possibile\n";
      return saldo;
    }
    else return saldo -= p;
  }
```

```
};


class ContoArancio: public ContoDiRisparmio {
  private:
  ContoCorrente& capp; //conto di appoggio
  public:
  ContoArancio(ContoCorrente& c): capp(c) {}

  int deposita(int d) { capp.preleva(d); return saldo += d; }
  int preleva(int p) {
    if (p > saldo) {
      cout << "Il prelievo non e' possibile\n";
      return saldo;
    }
    else {
      capp.deposita(p);
      return saldo -= p;
    }
  }
};
```