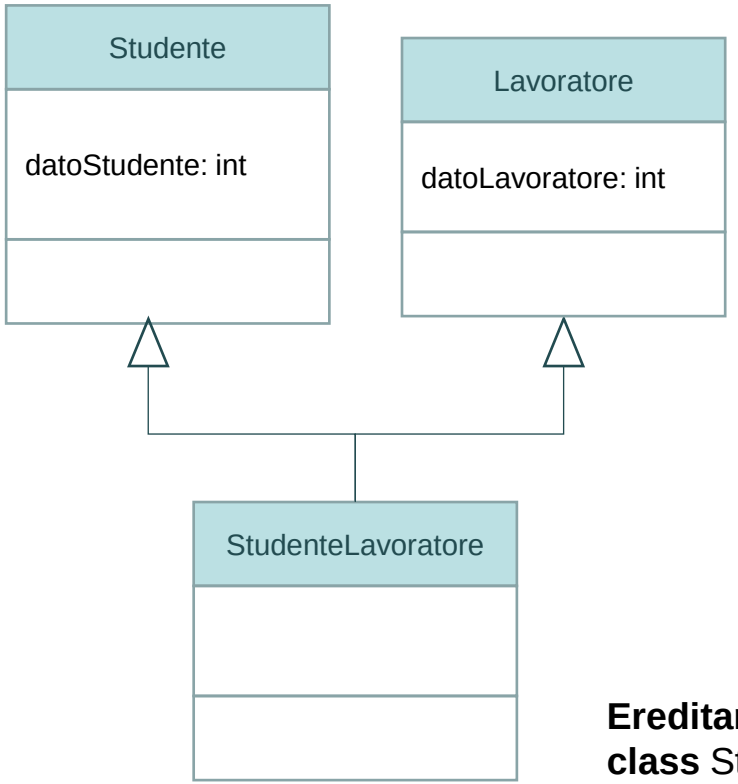
Programmazione Avanzata

Ereditarietà multipla

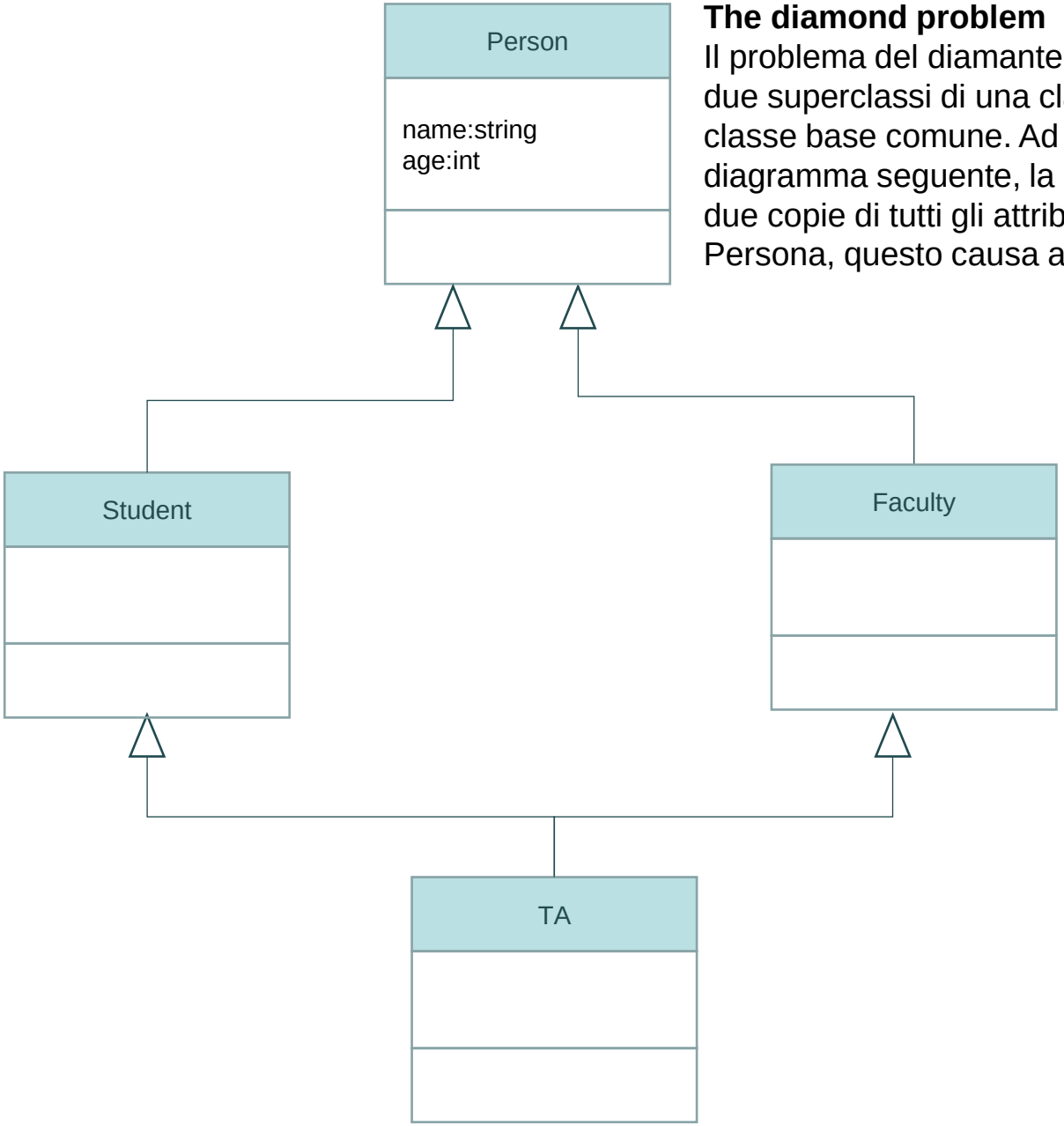
Pierluigi Roberti

pierluigi.roberti@unitn.it



Ereditarietà multipla

```
class StudenteLavoratore: public Studente, public Lavoratore{
        //….
};
```



The diamond problem

Il problema del diamante si verifica quando due superclassi di una classe hanno una classe base comune. Ad esempio, nel diagramma seguente, la classe TA ottiene due copie di tutti gli attributi della classe Persona, questo causa ambiguità.

```
class Person {
   // Data members of person
public:
    Person(int x)  { cout << "Person::Person(int ) called" << endl;   }
};

class Faculty : public Person {
   // data members of Faculty
public:
    Faculty(int x):Person(x)   {
       cout<<"Faculty::Faculty(int ) called"<< endl;
    }
};

class Student : public Person {
   // data members of Student
public:
    Student(int x):Person(x) {
        cout<<"Student::Student(int ) called"<< endl;
    }
};

class TA : public Faculty, public Student  {
public:
    TA(int x):Student(x), Faculty(x)   {
        cout<<"TA::TA(int ) called"<< endl;
    }
};
```

```
Person::Person(int ) called
Faculty::Faculty(int ) called
Person::Person(int ) called
Student::Student(int ) called
TA::TA(int ) called
```

```
class Person {
   // Data members of person
public:
    Person(int x)  { cout << "Person::Person(int ) called" << endl;   }
    Person()     { cout << "Person::Person() called" << endl;   }
};

class Faculty : virtual public Person {
   // data members of Faculty
public:
    Faculty(int x):Person(x)   {
       cout<<"Faculty::Faculty(int ) called"<< endl;
    }
};

class Student : virtual public Person {
   // data members of Student
public:
    Student(int x):Person(x) {
        cout<<"Student::Student(int ) called"<< endl;
    }
};

class TA : public Faculty, public Student  {
public:
    TA(int x):Student(x), Faculty(x)   {
        cout<<"TA::TA(int ) called"<< endl;
    }
};
```

Person::Person(int) called
Faculty::Faculty(int) called
Student::Student(int) called
TA::TA(int) called

Nel programma precedente, il costruttore di "Person" viene chiamato una volta.
Una cosa importante da notare nell'output precedente è il costruttore predefinito di "Persona« che viene chiamato.
Quando usiamo la parola chiave "virtual", il costruttore predefinito della classe non viene chiamato per impostazione predefinita anche se le classi padre chiama esplicitamente il costruttore parametrizzato.
Come chiamare il costruttore parametrizzato della classe "Person"?
Il costruttore deve essere chiamato nella classe "TA".
Ad esempio, vedere il seguente programma.

```
TA(int x):Student(x), Faculty(x), Person(x) {
cout<<"TA::TA(int ) called"<< endl;
}
```

In generale, non è consentito chiamare direttamente il costruttore del parent, deve essere chiamato tramite la classe genitore.
È consentito solo quando viene utilizzata la parola chiave "virtual".