IIT Bombay

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Inheritance in C++

Overview of This Lecture

- Inheritance
  - Motivation
  - Compositional vs. inheritance-based approaches
  - Hierarchy of classes

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

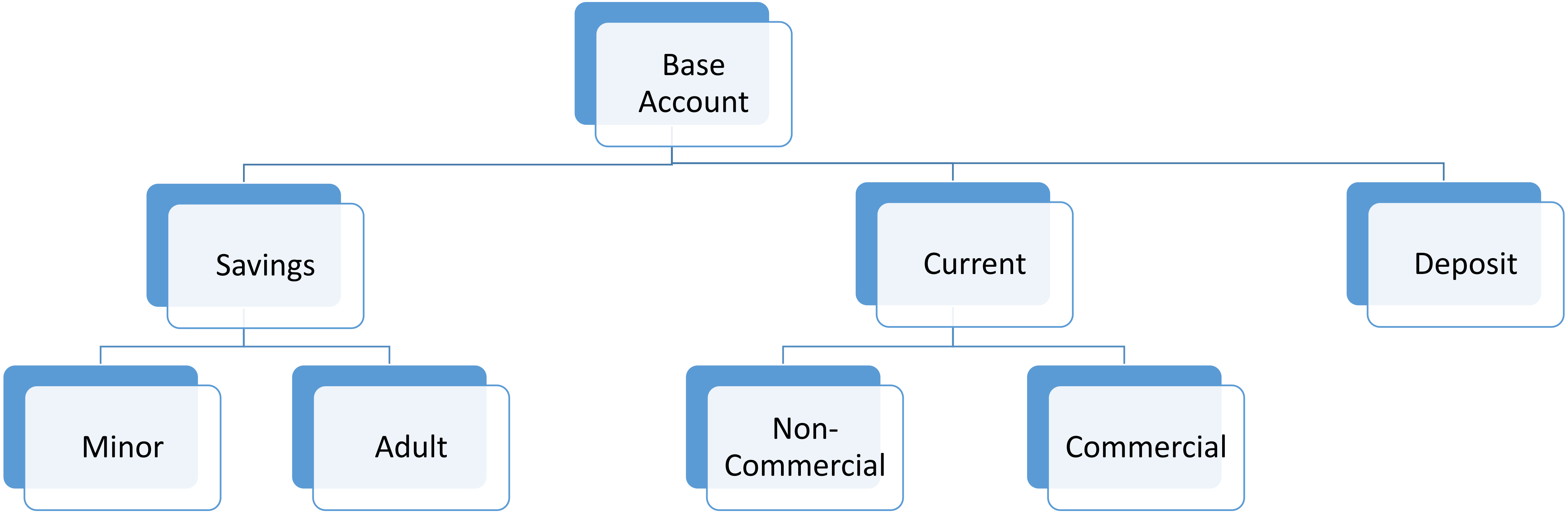
  by Abhiram G. Ranade

  McGraw Hill Education 2014

A Bank Account Example

IIT Bombay

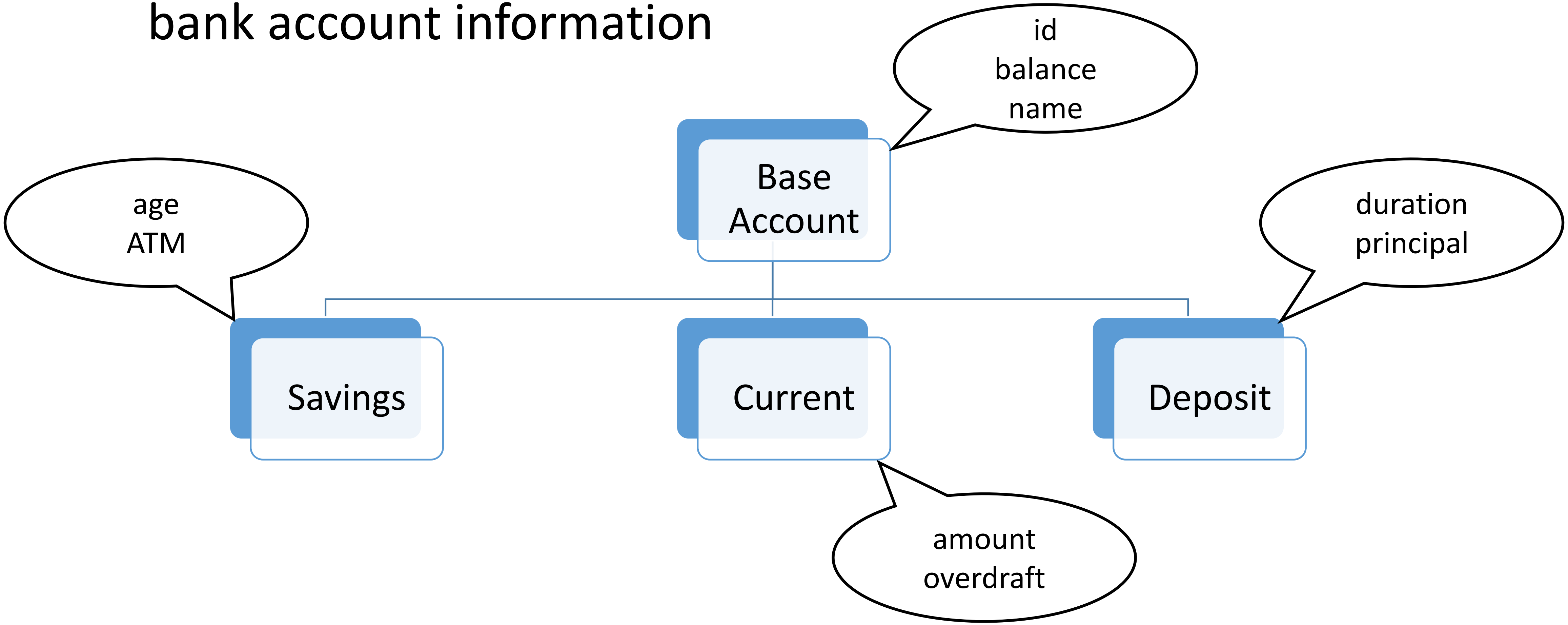
- Hierarchy of Accounts



A Bank Account Example

IIT Bombay

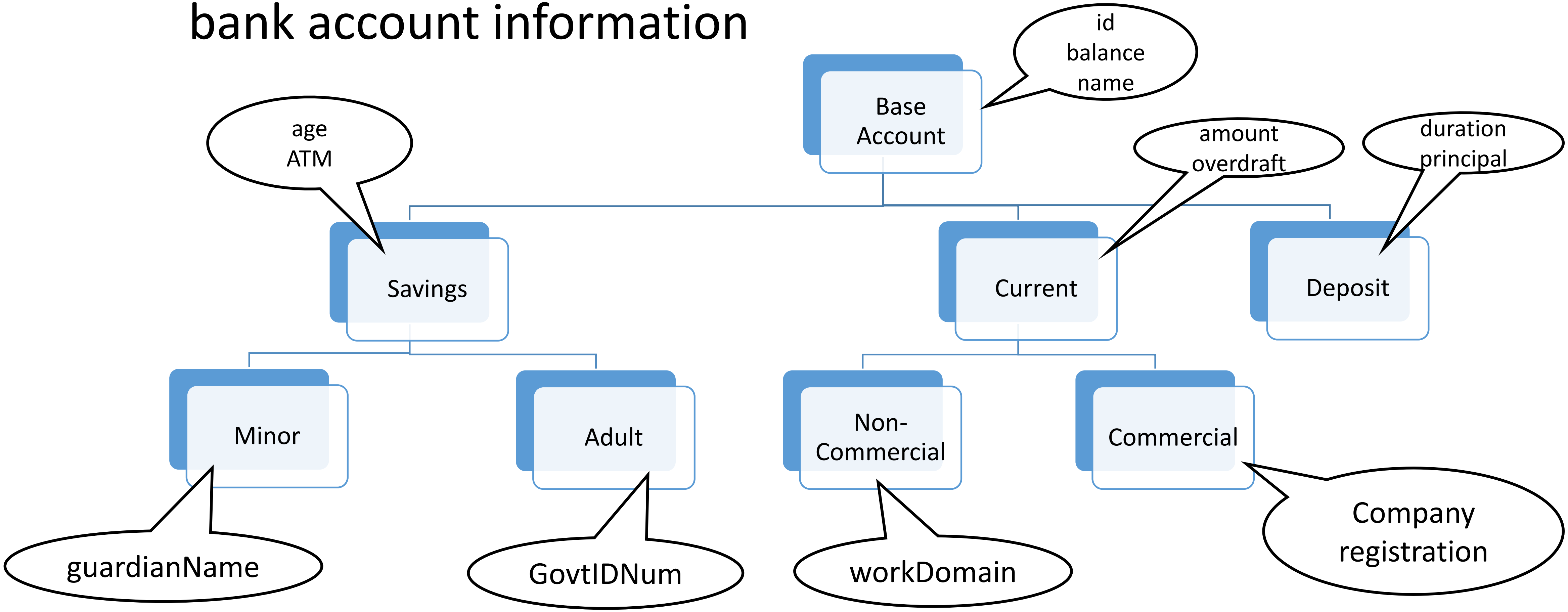
- We want to define a hierarchy of classes for representing bank account information



A Bank Account Example

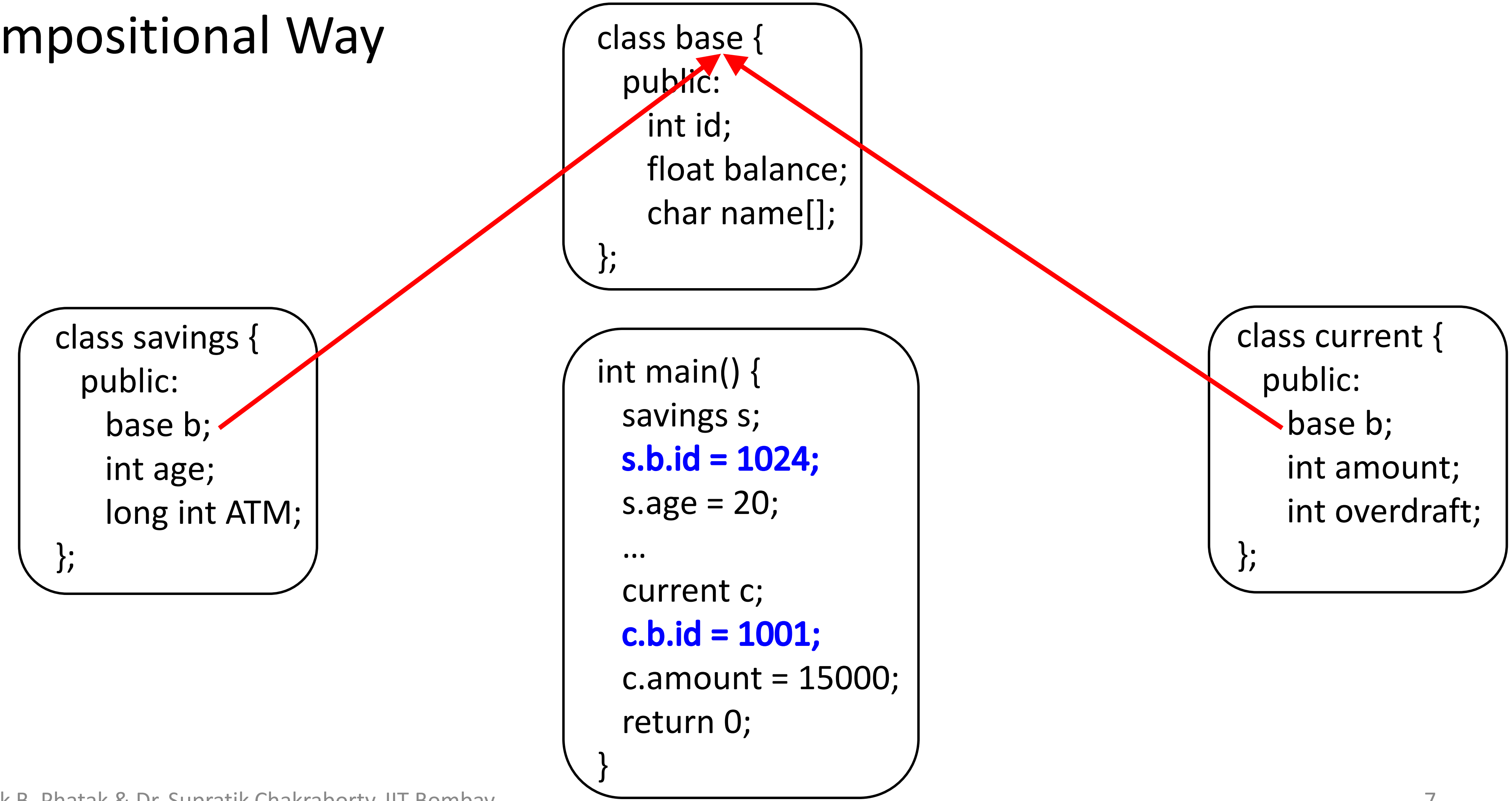
IIT Bombay

- We want to define a hierarchy of classes for representing bank account information



How to define these Classes ?

- Compositional Way



```
class base {
  public:
    int id;
    float balance;
    char name[];
};
```

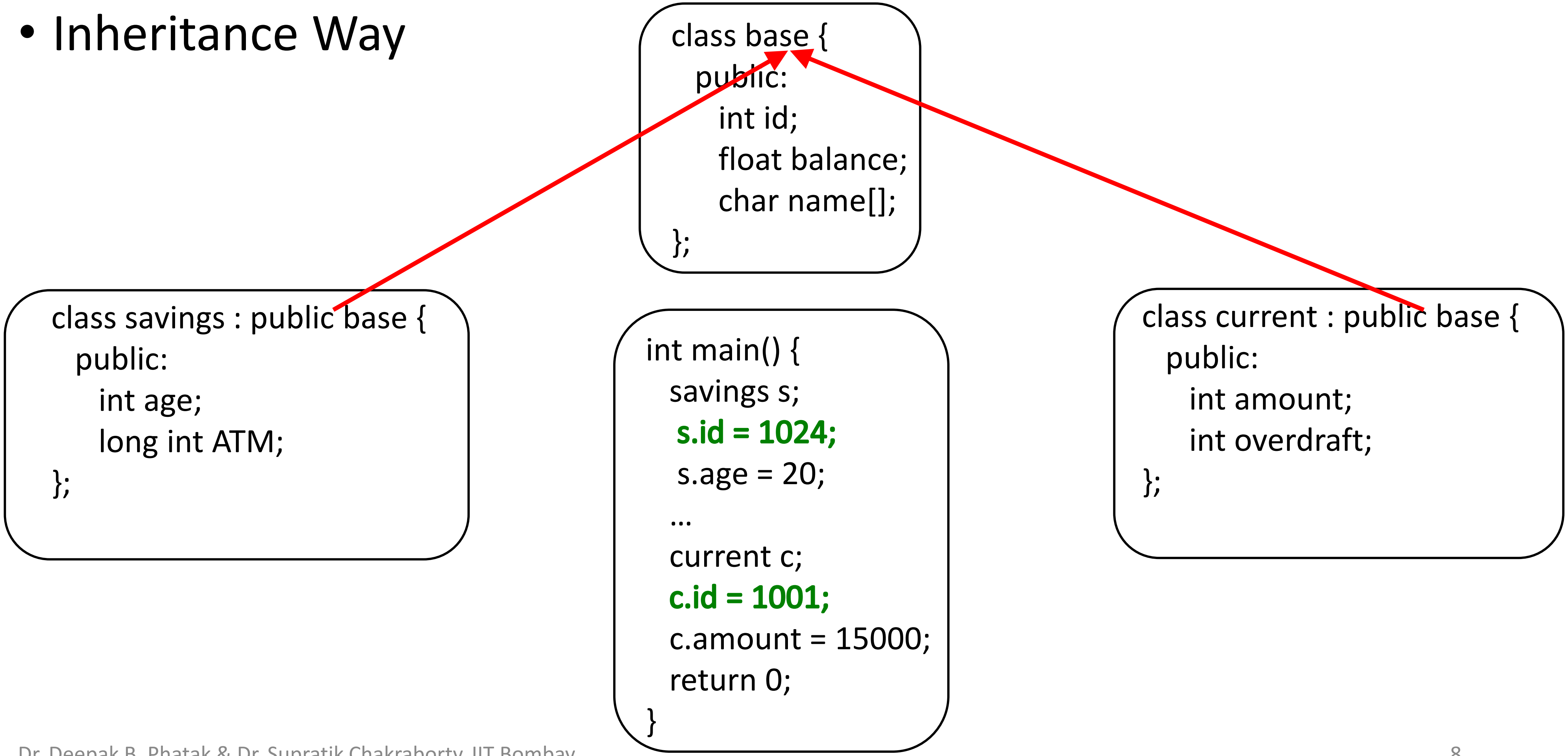
```
class savings {
  public:
    base b;
    int age;
    long int ATM;
};
```

```
int main() {
  savings s;
  s.b.id = 1024;
  s.age = 20;
  …
  current c;
  c.b.id = 1001;
  c.amount = 15000;
  return 0;
}
```

```
class current {
  public:
    base b;
    int amount;
    int overdraft;
};
```

How to define these Classes ?

- Inheritance Way



```
class base {
  public:
    int id;
    float balance;
    char name[];
};
```

```
class savings : public base {
  public:
    int age;
    long int ATM;
};
```

```
int main() {
  savings s;
   s.id = 1024;
   s.age = 20;
  …
  current c;
  c.id = 1001;
  c.amount = 15000;
  return 0;
}
```

```
class current : public base {
  public:
    int amount;
    int overdraft;
};
```

Difference in Access Style

Compositional Way

```
int main() {
   savings s;
   s.b.id = 1024;
   s.age = 20;
   …
   current c;
   c.b.id = 1001;
   c.amount = 15000;
   return 0;
}
```

Inheritance Way

```
int main() {
   savings s;
    s.id = 1024;
    s.age = 20;
   …
   current c;
   c.id = 1001;
   c.amount = 15000;
   return 0;
}
```

Complex Access Control

Requirement:

Members ‘id’ & ‘balance’ of class ‘base’

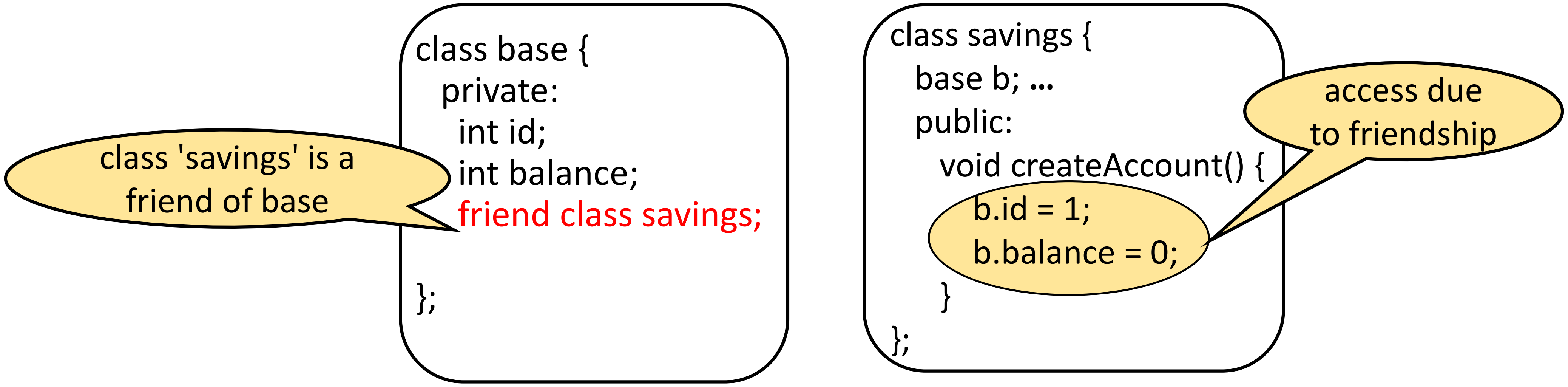
to be accessible

only from class ‘savings’,

and not from other classes

How do we achieve this?

Compositional: Private Member & Friend Class



```
class base {
   private:
     int id;
     int balance;
     friend class savings;

};
```

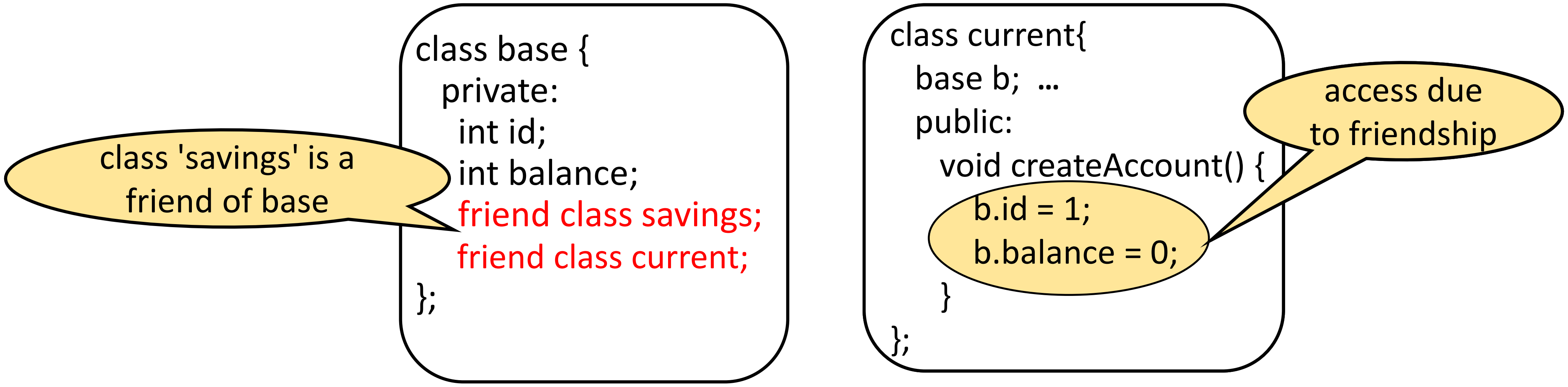
class 'savings' is a friend of base

```
class savings {
   base b; …
   public:
      void createAccount() {
         b.id = 1;
         b.balance = 0;
      }
};
```

access due to friendship

Compositional: Private Member & Friend Class

To extend this privilege to ‘current’ class



```
class base {
   private:
     int id;
     int balance;
     friend class savings;
     friend class current;
};
```

class 'savings' is a friend of base

```
class current{
   base b;  …
   public:
      void createAccount() {
         b.id = 1;
         b.balance = 0;
      }
};
```

access due to friendship

Can we do without inserting “friend class” declarations every time?

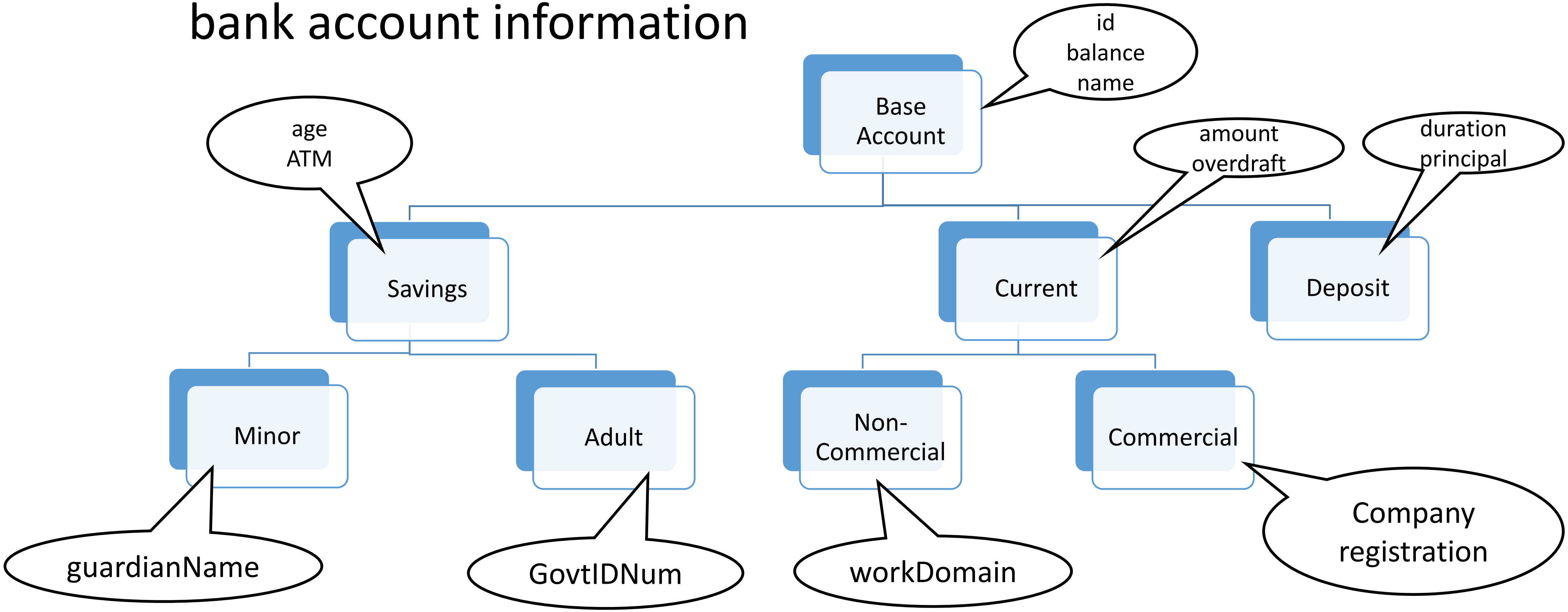
Towards Derived Classes

Can we explicitly say that a class is **derived** from a **base class** and **inherits attributes** of base class with special privileges to access those attributes?

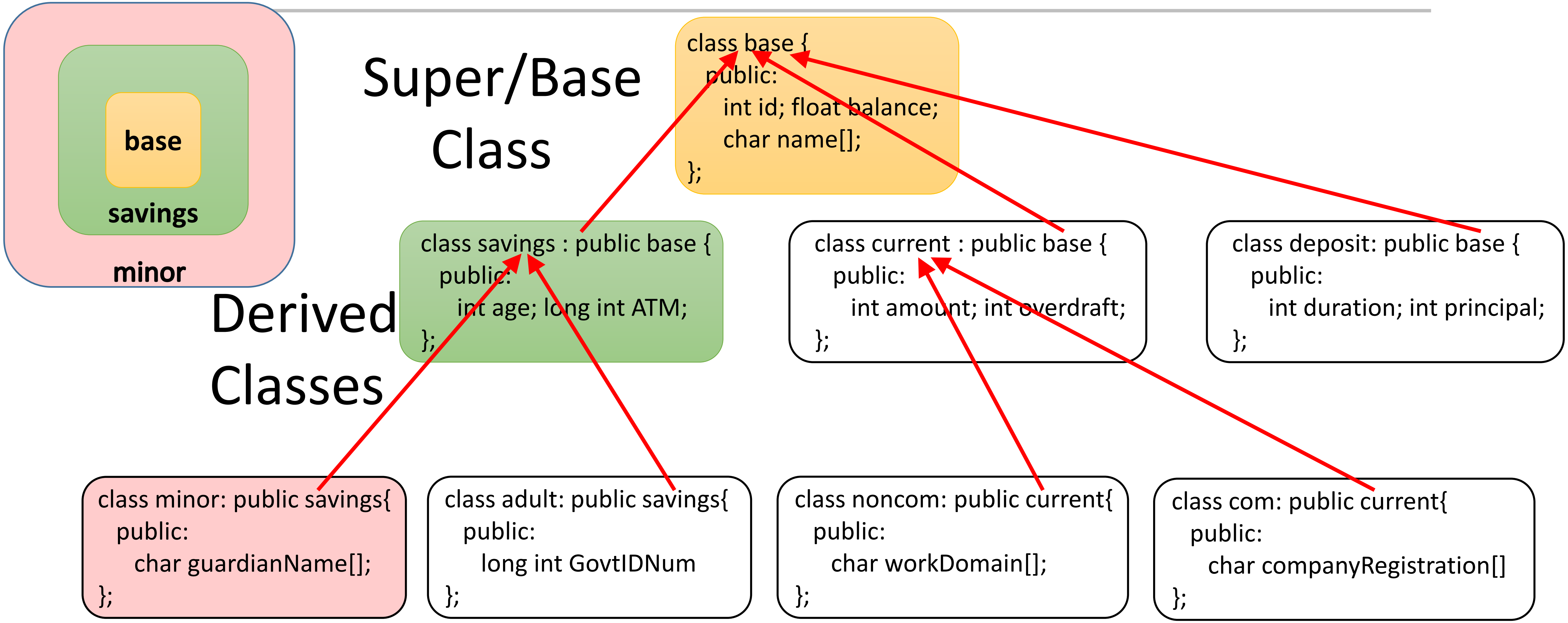
Inheritance in C++

How do we specify Inherited Classes

- We want to define a hierarchy of classes for representing bank account information



Class Hierarchy



Super/Base Class

```
class base {
  public:
    int id; float balance;
    char name[];
};
```

Derived Classes

```
class savings : public base {
  public:
    int age; long int ATM;
};
```

```
class current : public base {
  public:
    int amount; int overdraft;
};
```

```
class deposit: public base {
  public:
    int duration; int principal;
};
```

```
class minor: public savings{
  public:
    char guardianName[];
};
```

```
class adult: public savings{
  public:
    long int GovtIDNum
};
```

```
class noncom: public current{
  public:
    char workDomain[];
};
```

```
class com: public current{
  public:
    char companyRegistration[]
};
```

Complete Program

A complete program which uses all public members and shows how to access members of base class from derived class

```
#include<iostream>
using namespace std;

class base {
   public:
   int id;
   float balance;
   char name[];
};
```

```
class savings : public base {
   public:
   int age;
   long int ATM;
};

class current : public base {
   public:
   int amount;
   int overdraft;
};
```

```
int main() {
   savings s;
   s.id = 1;
   s.age = 20;
   cout << s.id << s.age;
   current c;
   c.id = 2;
   c.amount = 15000;
   cout << c.id << c.amount;
   return 0;
}
```

Summary

- Compositional vs. inheritance-based approaches
- Defining hierarchy of classes