9. Inheritance: Concept, Issues & Implementation

Inheritance represents “is-a” or “kind-of” relationship.

- Reusability
- Extensibility
- Efficiency

Base class & Derived class
(super Class & Sub Class)
Example of Inheritance

```cpp
class Employee {
    char* name;
    char* title;
public:
    Employee (char* nme, char* ttl);
    virtual void print() {
        cout << name << " " << ttl; }
};
void SecurityCheck(const Employee&);
Employee bruce, joe, kim;
Employee* eptr = &bruce;
```

```cpp
class Manager : public Employee {
    int level;
    Employee* manages_list;
public:
    Manager(char* nme, char* ttl, int lev);
    virtual void print();
};
ManagersConference(const Manager&);
Manager bob;
Manager* mptr = &bob;
```

Example of Inheritance... cont.

```cpp
Employee bruce;
Employee* eptr;
SecurityCheck(bruce);
SecurityCheck(bob);
Manager bob;
Manager* mptr;
mptr = &bob;
eptr = &bob;
```
Conversions

Which of the following are valid.

Employee bruce; Employee* eptr;
Manager bob; Manager* mptr;

eptr = &bruce; ✔
eptr = &bob; ✔
eptr = mptr; ✔
mptr = &bob; ✔
mptr = &bruce; ✗
mptr = eptr; ✗

Conversions between Base & Derived

Base* bptr; Derived* dptr;

bptr = dptr; //OK
dptr = bptr; // ERROR

Conversion from Base to Derived possible by Explicit conversion:

mptr = (Manager*) eptr; //OK --- CAUTION

Use RTTI to verify type before conversion, if you have to.
Dynamic Cast

• If a type conversion can not be validated at compile time, it is safer to use Dynamic Cast to validate it at run time.

• dynamic_cast will cast a pointer to
  – an appropriate pointer type if legal
  – 0 otherwise

• Type* ptr = dynamic_cast<Type*> (somePointer);
• if (ptr != 0) somePointer is pointing to an object of type “Type”.

Dynamic cast on reference

• Pointer may be null.
• Reference can not be null

• dynamic_cast of reference is more of an assertion than a query

try
{
  Type& myRef = dynamic_cast<Type&> (anotherRef);
}
catch(bad_cast) { // Assertion failed … }

• if anotherRef is not referring to an object of type “Type”, then bad_cast exception thrown
Writing the Constructor of Derived class

Manager:: Manager(char* nme, char* ttl, int lev) :
  Employee (nme, ttl)
{
  level = lev;
}

Order of Construction & Destruction

List the order of creation (calls to constructors) of the following objects:
Analyst sam; _________________________________________
Manager nancy; _________________________________________
Directory laura; _________________________________________
Abstract Base Class

Representing an Abstraction that is Abstract.
• Abstract classes represent concepts not real objects
• ABCs are used only to create other “Concrete” classes

Example: Shape

ABCs are implemented in C++ using pure Virtual Functions

Example of an ABC

Employee may be an ABC

class Employee{...
virtual void YearlyReview() = 0; // Pure Virtual Function
};

class Analyst : public Employee{...
virtual void YearlyReview()
{
    // Procedure for the Yearly Review for an Analyst.
}
};
Lab Work: Details provided on-line.