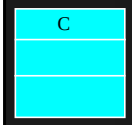
Coding

145

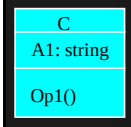
C++ : Class




```
Class C {
public:
   // constructors & destructors
   C();
   ~C();
   // assignment and equality operators
   const C& operator = (const C& right);
   int operator == (const C& right) const;
   int operator != (const C& right) const;
};
```

146

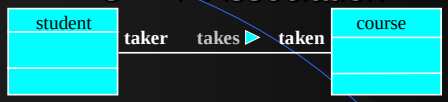
C++ : Class with Attributes & Op.




```
class C {
private:
   String A1;
public:
   ...
   void Op1();
   const String get_A1() const;
   void set_A1(const String value);
}
const String C:: get_A1() const {
   return A1;                    //  ..........
```

147

C++ : Association

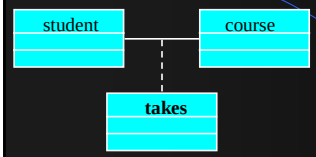



```
class student {
private:
   course*  taken;
public:
   const course* get_taken()
   const;
   void set_taken(B* const
                  value);
}
```

```
class course {
private:
   student*  taker;
public:
   const student* get_taker()
   const;
   void set_taker(A* const
                  value);
}
```

148

C++ : Association Class




```
class takes {
private:
   student*  the_student;
   course*  the_course;
   ............}
```

```
class student {
private:
   takes*  the_takes;
.....  }
```

```
class course {
private:
   takes*  the_takes;
............}
```

149

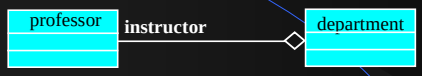
C++ : Composition


```
class wing {
private:
   pinguin*  thepinguin;
   .......
}
```

```
class pinguin {
private:
   wing flipper;
   ........
}
```

150

C++ : Aggregation

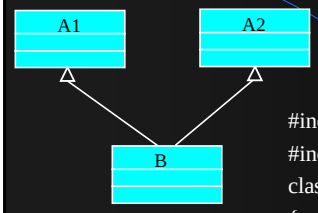



```
class professor {
private:
   department*  thedepartment;
  .......
}
```

```
class department {
private:
    professor*  instructor;
   ........
}
```

151

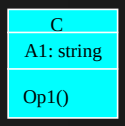
C++ : Inheritance




```
#include "A1.h"
#include "A2.h"
class B : public A2,public A1
{
  .......
}
```

152

Java: Class

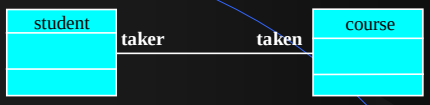



```
public final class C {
   private String m_A1;
   public void Op1() {
         ....
   }
   ......
}
```

[final | abstract | interface !]

153

Java : Association

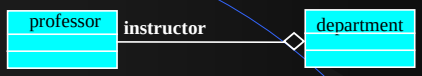



```
public class student {

Public
        course  m_taken;
   .....
}
```

```
public class course {

Public
        student  m_taker;
   .....
}
```

154

Java: Aggregation

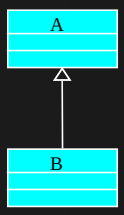



```
class professor {
public department
   m_department;
  .......
}
```

```
class department{
public professor
   m_instructor;
  ........
}
```

155

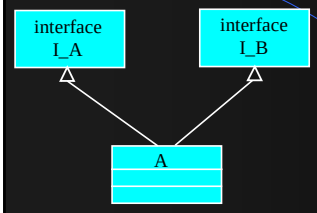
Java : Single Inheritance




```
public class B extends A
{
  .......
}
```

156

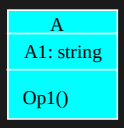
Java : “implementing” interfaces




```
public class A implements I_A, I_B
{
   .......
}
```

157

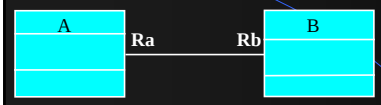
SQL : Class




```
CREATE TABLE T_A (
   A_Id NUMBER  (5),
   A1 VARCHAR (),
   PRIMARY KEY (A_Id)
)
```

158

SQL : Association

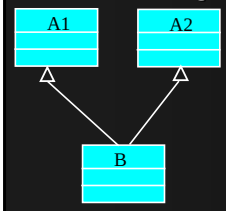



```
CREATE TABLE T_B (
  B_Id NUMBER (5)
  PRIMARY KEY (B_Id)
)
```

```
CREATE TABLE T_A (
  A_Id NUMBER (5)
  B_Id NUMBER (5) REFERENCES T_B (B_Id)
  PRIMARY KEY (A_Id)
)
```

159

SQL : Inheritance




```
CREATE TABLE T_A1 (
  A1_Id NUMBER (5)
  PRIMARY KEY (A1_Id)
)
CREATE TABLE T_A2 (
  A2_Id NUMBER (5)
  PRIMARY KEY (A2_Id)
)
```

```
CREATE TABLE T_B (
  A1_Id NUMBER (5) REFERENCES T_A1 (A1_Id)
  A2_Id NUMBER (5) REFERENCES T_A2 (A2_Id)
  PRIMARY KEY (A1_Id, A2_Id)
)
```

160