Object-Oriented Programming (OOP)
Lecture No. 38

Templates and Friends

- Like inheritance, templates or their specializations are compatible with friendship feature of C++

Templates and Friends – Rule 1

- When an ordinary function or class is declared as friend of a class template then it becomes friend of each instantiation of that template

…Templates and Friends – Rule 1

```
void doSomething( B< char >& );

class A { … };

template< class T > class B {
  int data;
  friend void doSomething( B<char>& );
  friend A;
  …
};
```

…Templates and Friends – Rule 1

```
void doSomething( B< char >& cb ) {
  B< int > ib;
  ib.data = 5;    // OK
  cb.data = 6;    // OK
}
```

…Templates and Friends – Rule 1

```
class A {
  void method() {
     B< int > ib;
     B< char > cb
     ib.data = 5; // OK
     cb.data = 6; // OK
  }
};
```

Templates and Friends – Rule 2

- When a friend function / class template is instantiated with the type parameters of class template granting friendship then its instantiation for a specific type is a friend of that class template instantiation for that particular type

…Templates and Friends – Rule 2

```
template< class U >
void doSomething( U );
template< class V >
class A { … };

template< class T > class B {
  int data;
  friend void doSomething( T );
  friend A< T >;
};
```

…Templates and Friends – Rule 2

```
template< class U >
void doSomething( U u ) {
  B< U > ib;
  ib.data = 78;
}
```

…Templates and Friends – Rule 2

```
int main() {
  int i = 5;
  char c = ‘x’;
  doSomething( i );   // OK
  doSomething( c );   // OK
  return 0;
}
```

…Templates and Friends – Rule 2

```
template< class U >
void doSomething( U u ) {
  B< int > ib;
  ib.data = 78;
}
```

…Templates and Friends – Rule 2

```
int main() {
  int i = 5;
  char c = ‘x’;
  doSomething( i );   // OK
  doSomething( c );   // Error!
  return 0;
}
```

…Templates and Friends – Rule 2

- Because `doSomething()` always instantiates `B< int >`

```
class B< int > {
  int data;
  friend void doSomething( int );
  friend A< int >;
};
```

…Templates and Friends – Rule 2

```
template< class T >
class A {
  void method() {      // Error!
     B< char > cb;
     cb.data = 8;
     B< int > ib;
     ib.data = 9;
  }
};
```

Templates and Friends – Rule 3

- When a friend function / class template takes different type parameters from the class template granting friendship then its each instantiation is a friend of each instantiation of the class template granting friendship

…Templates and Friends – Rule 3

```
template< class U >
void doSomething( U );
template< class V >
class A { … };
template< class T > class B {
  int data;
  template< class W >
     friend void doSomething( W );
  template< class S >
     friend class A;
};
```

…Templates and Friends – Rule 3

```
template< class U >
void doSomething( U u ) {
  B< int > ib;
  ib.data = 78;
}
```

…Templates and Friends – Rule 3

```
int main() {
  int i = 5;
  char c = 'x';
  doSomething( i );    // OK
  doSomething( c );    // OK
  return 0;
}
```

…Templates and Friends – Rule 3

```
template< class T >
class A {
  void method() {      // OK!
     B< char > cb;
     cb.data = 8;
     B< int > ib;
     ib.data = 9;
  }
};
```

Templates and Friends – Rule 4

- Declaring a template as friend implies that all kinds of its specializations – explicit, implicit and partial, are also friends of the class granting friendship

…Templates and Friends – Rule 4

```
template< class T >
class B {
  T data;
  template< class U >
     friend class A;
};
```

…Templates and Friends – Rule 4

```
template< class U >
class A {
  A() {
     B< int > ib;
     ib.data = 10;     // OK
  }
};
```

…Templates and Friends – Rule 4

```
template< class U >
class A< U* > {
  A() {
     B< int > ib;
     ib.data = 10;      // OK
  }
};
```

…Templates and Friends – Rule 4

```
template< class T >
class B {
  T data;
  template< class U >
     friend void doSomething( U );
};
```

…Templates and Friends – Rule 4

```
template< class U >
void doSomething( U u ) {
  B< int > ib;
  ib.data = 56;   // OK
}
```

…Templates and Friends – Rule 4

```
template< >
void doSomething< char >( char u ) {
  B< int > ib;
  ib.data = 56;   // OK
}
```