TEMPLATE CLASSES

Generalized Class Members

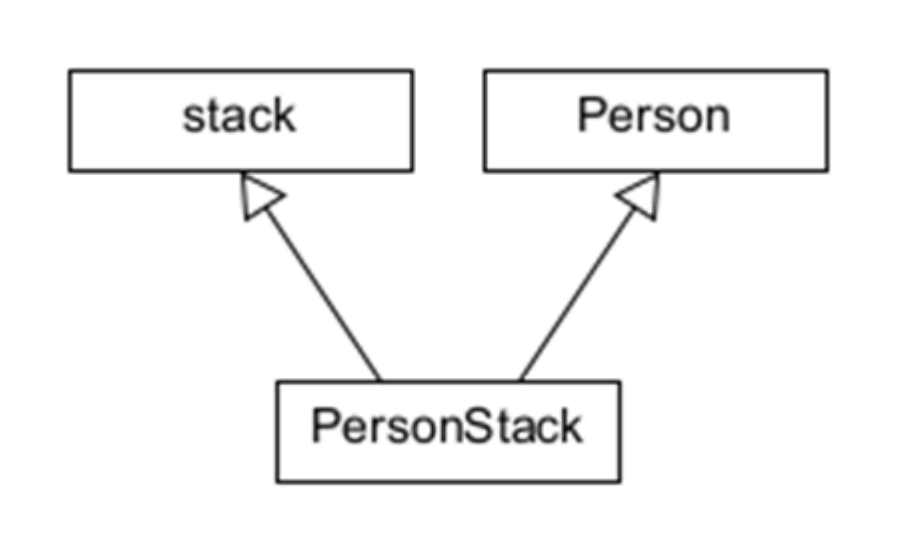
Delroy A. Brinkerhoff

ACTIVATING TEMPLATES AND INTRODUCING THE VARIABLE

- `template <class T>`
- “class” used in documentation
- Appropriate for template classes

- `template <typename T>`

DATA STRUCTURES



- Multiple inheritance
  - Data structure operations from stack
  - Data and operations from Person
- Template class
  - Data structure operations from stack
  - Data specified by application

SPECIFYING A TEMPLATE CLASS

The "template" statement and variable

```
template <class T>
class stack
{
    private:
        static const int SIZE = 100;

        T    st[SIZE];
        int  sp = 0;

    public:
        void push(T data);
        T    pop();
        int  size();
        T    peek();
};
```

TEMPLATE MEMBER FUNCTIONS

```
template <class T>
void stack<T>::push(T data)
{
    if (sp < SIZE)
        st[sp++] = data;
    else
        throw "Stack Overflow";
}
```

```
template <class T>
T stack<T>::pop()
{
    if (sp > 0)
        return st[--sp];
    else
        throw "Stack Underflow";
}
```

TEMPLATE MEMBER FUNCTIONS

```
template <class T>
int stack<T>::size()
{
    return sp;
}

template <class T>
T stack<T>::peek()
{
    return st[sp - 1];
}
```

```
int main()
{
    stack<Person>  p;
    Person         x("Alice");
    Person         y("Dilbert");

    p.push(x);
    p.push(y);

    p.pop().display();
    p.pop().display();

    return 0;
}
```

TEMPLATE CONSTANTS

WITHOUT A DEFAULT

```
template <class T, int SIZE>
class stack
{ ... };

template <class T, int SIZE>
void stack<T, SIZE>::push(T data)
{ ... }

stack<int, 10> s1;
```

WITH A DEFAULT

```
template <class T, int SIZE = 100>
class stack
{ ... };

template <class T, int SIZE>
void stack<T, SIZE>::push(T data)
{ ... }

stack<int, 10> s2;
stack<int> s3;
```