Object Oriented Programming Lecture

Inheritance I

School of Computing and Information Technology
Faculty of Engineering and Computing
University of Technology, Jamaica

Email: dwwhite@utech.edu.jm, taedwards@utech.edu.jm

Object Oriented Programming

Expected Outcome

At the end of this lecture the student should be able to:

- Explain how inheritance facilitates software reuse
- Identify candidates for inheritance relationships during object-oriented analysis
- Represent inheritance relationships in UML
- Represent inheritance relationships in code

Object Oriented Programming

Topics to be covered in this lecture:

- Software reuse through inheritance
- Base and derived classes
- Single and multiple Inheritance
- How access specifiers affect inheritance

Object Oriented Programming

Topics to be covered in this lecture:

- Identifying inheritance relationships during object-oriented analysis
- Inheritance relationships in UML
- Inheritance relationships in code

Inheritance: Software Reuse

- Inheritance describes a relationship in which child classes derive members from their parent classes
- Inheritance is a key concept of OOP
- The state (attributes) and the behaviours (methods) of the parent class are inherited by the child class
- Can be used to eliminates redundant code
- Promotes software reuse

Inheritance: Software Reuse

Inheritance promotes software reuse through:

- Child classes inherit the code already written for their parents
- General attributes and methods that apply to all the children are placed in the parent class
- Once a parent class has been written and tested, it may be inherited as often as needed by children
- Each child class may be modified or specialized without affecting the parent class

Inheritance: Software Reuse

Inheritance, through software reuse:

- Reduces software development time
  - Less time is spent writing code
- Reduces software development cost
  - Less code implies less cost
- Increases software reliability
  - If tried, tested and proven parent classes are reused through inheritance, then child classes will inherit the same tried, tested, proven code.

Inheritance: Base and Derived classes

- The general members that apply to a set of classes in an inheritance relationship are placed in the parent class called the base
- Hence, the parent is called a generalized class
- Each child class inherits from the parent class and may customize itself to differentiate itself from other child classes, hence it is called a specialization or derived class
- Other synonyms exist for parent and child

Inheritance: Base and Derived classes

Base Class Synonyms	**Derived Class Synonyms**
Generalized	**Specialized**
Parent	Child
Super	Sub

The terms in the table are used together, so for example, one can speak of parent and child classes, or base and derived classes.

Inheritance: Single vs Multiple

- Single inheritance occurs when a child inherits from **only one parent**
- Multiple Inheritance occurs when a child inherits from **more than one parent**

Inheritance: Single vs Multiple

- Multiple inheritance can cause problems with ambiguity if the same method name is inherited from more than one parent (which one will the child use?)
- C++ supports single and multiple inheritance
- Java only directly supports single inheritance but ***interfaces*** can simulate multiple inheritance

Inheritance: Role of Access Specifiers

- **All** members of a parent class are inherited by a child class, **except** for the constructors and destructor
- However, access specifiers can determine if the child class has access to the inherited members
- Access specifiers: private, public and protected

Inheritance: Role of Access Specifiers

Private

- The members of the parent class are inherited but cannot be accessed by the child class

Public

- The members of the parent class are inherited and can be accessed by the child class and other classes

Protected

- The members of the parent class are inherited and can only be accessed by the child its descendants

Inheritance: Identifying the Relationship

- When conducting Object-Oriented Analysis (OOA), terms such as “is a” hint at a possible inheritance relationship
- For this reason, inheritance is also called an “is-a” relationship or a generalization

Inheritance: Identifying the Relationship

- For example: A library has books. A text book is a book that covers a particular subject area. A dictionary is a book that contains definitions for a list of words.
- Book would be the base class, while dictionary and text book would be child classes of book

Inheritance: Identifying the Relationship

If a set of classes are very similar, containing common attributes and/or methods:

- Define a parent class
- Place the common attributes and methods in the parent class
- Place only the attributes and methods that make each child class unique in their respective class

Inheritance: Identifying the Relationship

Example:

A dialog box has a title, x and y coordinates, height, and width, and some message text. It has click close, click maximize, click minimize and click ok operations. A modal form has a title, warning level and some message text. It has click ok and beep operations.

- Perform an OOA on the above to identify the classes, attributes, methods and relationship

Inheritance: Identifying the Relationship

Class: dialog box

- Attributes: title, x coordinate, y coordinate, height, width, and message text
- Methods: clickClose, clickMaximize, clickMinimize, and clickOk

Class: modal form

- Attribute: title, warning level, and message text
- Method: clickOk, and beep

Inheritance: Identifying the Relationship

Class: window

- Attributes: title, and message text
- Method: clickOk

Inheritance: Identifying the Relationship

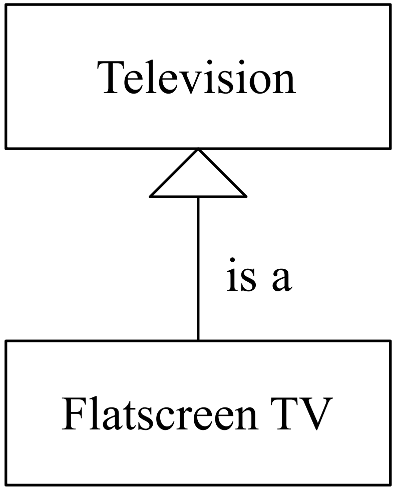
Parent class: window

Child classes: modal form, dialog box

- Classes modal form and dialog box inherit the title and message text attributes and the clickOk method from the window class.

Inheritance: Relationships in UML

In UML, an inheritance (“is-a” or generalization) relationship is depicted by a solid line connecting the parent and child classes. A hollow triangle (arrow head) is placed on the side of the line connecting the parent class



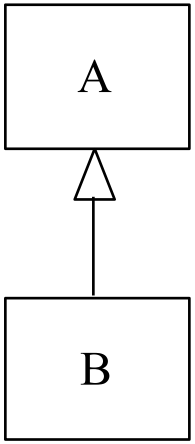
In this example, Flatscreen TV is a Television

Flatscreen TV is the child class and Television is the parent class

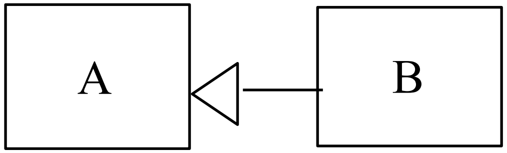
Flatscreen TV inherits the attributes and methods of Television

Inheritance: Relationships in UML

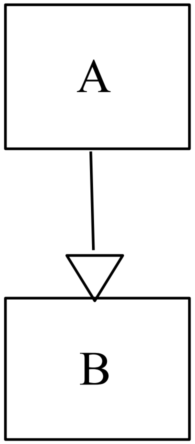
Single inheritance



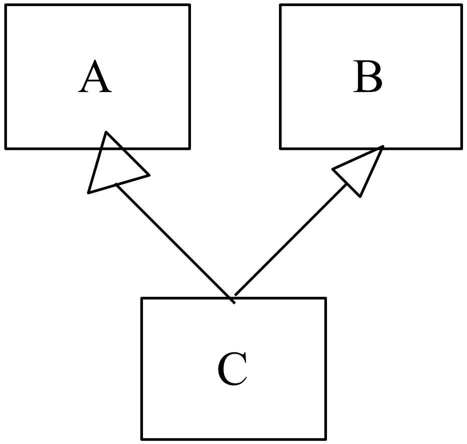
Single inheritance



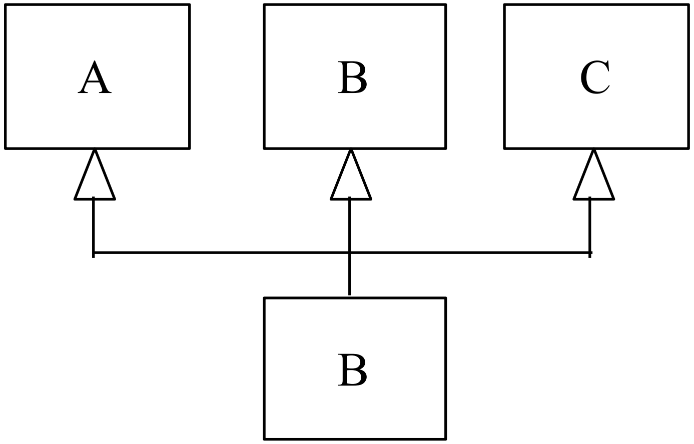
Single inheritance



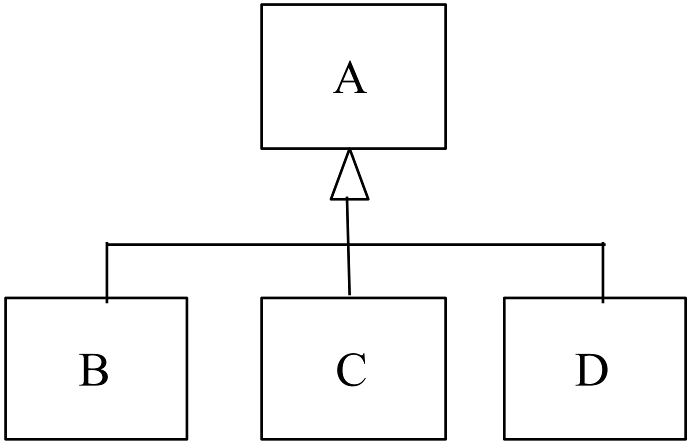
Multiple inheritance



Multiple inheritance

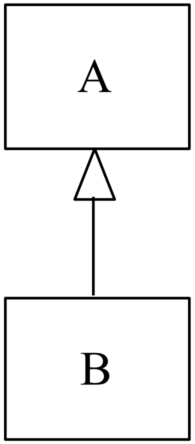


Single inheritance

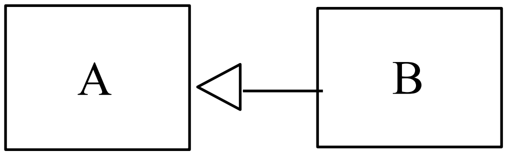


Inheritance: Relationships in UML

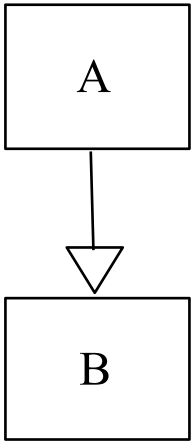
B inherits from A



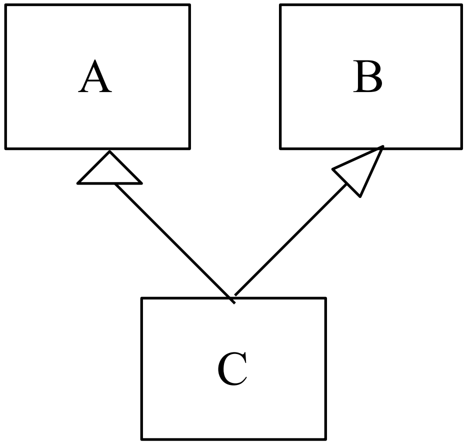
B inherits from A



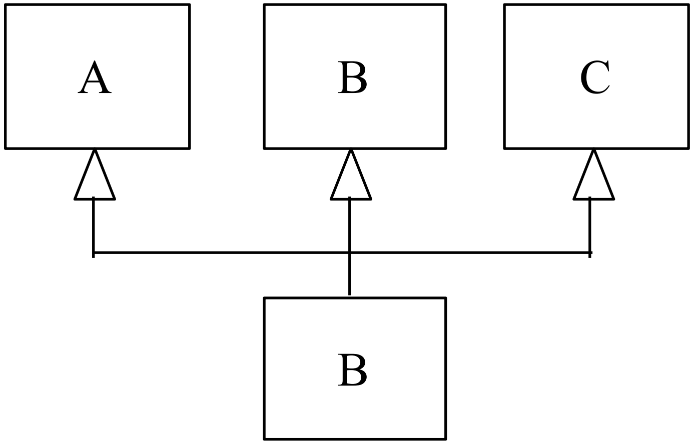
A inherits from B



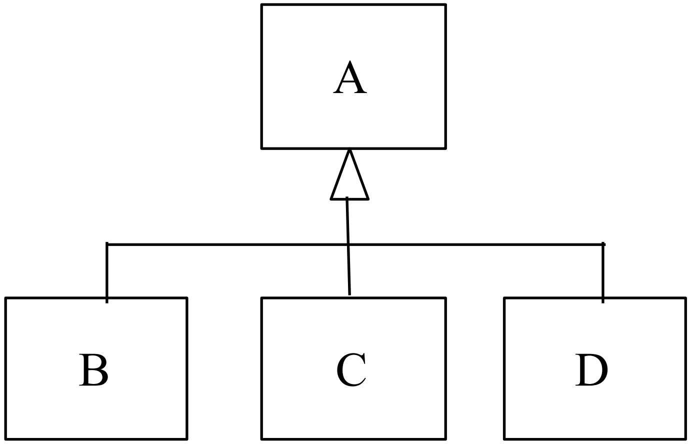
C inherits from A and B



B inherits from A, B and C

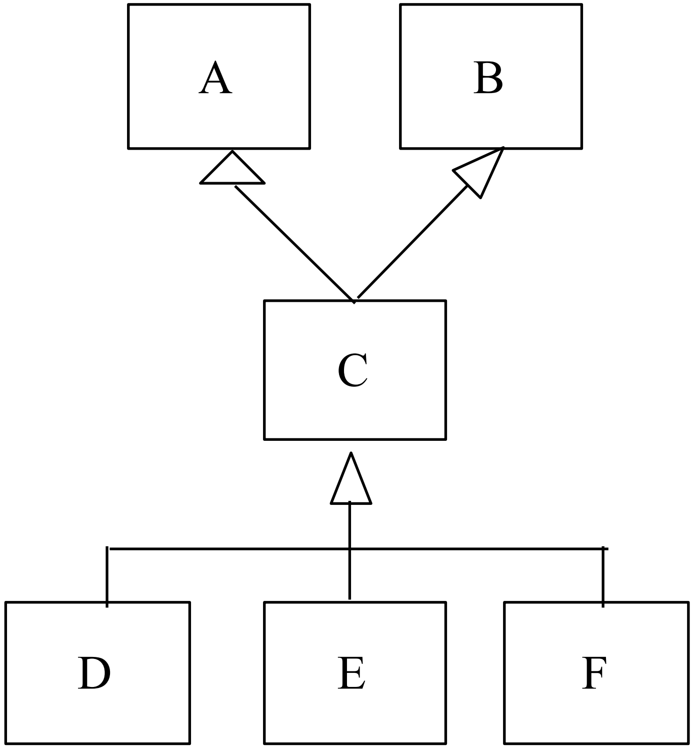


B, C and D inherits from A



Inheritance: Relationships in UML

An Inheritance Hierarchy



A and B are both parents and grandparents. They are direct super classes of C, and indirect super classes of D, E and F

C is a child class (sub class) of A and B, but C is also the parent class (super class) of D, E and F

D, E and F are child classes of C which means they are also children of A and B. Therefore D, E and F inherit all the members of A, B and C

Inheritance: Relationships in code

Java

If class B inherits from class A:

- Import the package with the class containing containing A

import pkga.A;

- Place the keyword ***extends*** after the name of the child class followed by the name of the parent class

- *public class B extends A {*

C++

If class B inherits from class A:

- Include the header file containing A

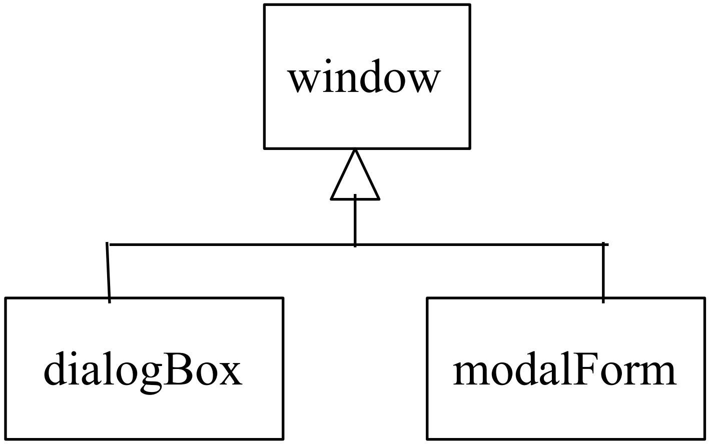
#include “A.h”

- Place a ***colon*** after the name of the child class followed by ***public*** followed by the name of the parent class

- *class B : public A {*

Inheritance: Relationships in code

A simple example showing how inheritance is implemented in C++ and Java, based on these UML diagrams



window

#title : string
#messageText : string

+clickOk(): void

modalForm

-warningLevel

+beep(): void

dialogBox

-x : int
-y : int
-height : int
-width : int

+clickClose() : void
+clickMaximize() : void
+clickMinimize() : void

The access specifiers in the UML are

·private	-	(minus sign)
·public	+	(plus sign)
·protected	#	(number sign)

A default constructor will be added to the child classes to show how they can access the inherited attributes

Inheritance: Relationships in code

C++

```
//Window.h
#include <iostream>
#include <string>
using namespace std;

class Window {

		protected:
			string title;
			string messageText;

		public:
			void clickOk()
			{
				cout << "OK was 
clicked" << endl;
			}
};
```

Java

```
//Window.java
public class Window {

	protected String title;
	protected String 
messageText;

	public void clickOk()
	{

	System.out.println("OK 
was clicked");
	}
}
```

Inheritance: Relationships in code

C++

```
//DialogBox.h
#include “Window.h"
class DialogBox : public Window {
        private:
                int x;
                int y;
                int height;
                int width;
        public:
                void clickClose()
                {}
                void clickMaximize()
                {}
                void clickMinimize()
                {}
                DialogBox()
                {
                        x = 0;
                        y = 0;
                        height = 0;
                        width = 0;
                        title = "";
                        messageText = "";
                }
};
```

Java

```
//DialogBox.java
public class DialogBox extends Window {
        private int x;
        private int y;
        private int height;
        private int width;

        public void clickClose()
        {}
        public void clickMaximize()
        {}
        public void clickMinimize()
        {}
        public DialogBox()
        {
                x = 0;
                y = 0;
                height = 0;
                width = 0;
                title = "";
                messageText = "";
        }
}
```

Inheritance: Relationships in code

C++

```
//ModalForm.h
#include "Window.h"
class ModalForm : public Window
{
        private:
                int warningLevel;

        public:
                void beep()
                {}

                ModalForm()
                {
                        warningLevel = 0;
                        title = "";
                        messageText = "";
                }
};
```

Java

```

//ModalForm.java
public class ModalForm extends
Window
 {
        private int warningLevel;

        public void beep()
        {}

        public ModalForm()
        {
                warningLevel = 0;
                title = "";
                messageText = "";
        }
}
```

Inheritance: Relationships in code

C++

```
//driver.cpp
#include “DialogBox.h”
#include “ModalForm.h”

int main ()
{
                DialogBox db;
                db.clickMaximize();
                db.clickOK();

                ModalForm mf;
                mf.beep();
                mf.clickOk();


                return 0;
}
```

Java

```
//driver.java
public  class Driver
{

        public  static void main(String[]  args)
        {

                DialogBox  db = new
DialogBox();
                db.clickMaximize();
                db.clickOk();

                ModalForm  mf = new
ModalForm();
                mf.beep();
                mf.clickOk();
        }
}
```