System Analysis and Design

Introduction: Object Oriented Design

Salahaddin University
College of Engineering
Software Engineering Department
2011-2012

Amanj Sherwany
http://www.amanj.me/wiki/doku.php?id=teaching:su:system_analysis_and_design

What is discussed?

- Objects, state, behaviour and identity
- Classes
- Encapsulation
- Inheritance
- Polymorphism
- Static and dynamic binding
- Composition and aggregation

Recap

- There is no single definition of object-oriented programming.
- Some herald the bundling of state and behaviour as the key concept, some emphasise message passing between objects, etc.
- An often mentioned triple of key concepts are encapsulation, inheritance and dynamic binding.
- You are invited to form your own understanding of OOP paradigm.

Objects

- Objects have: *state*, *behaviour* and *identity*.
- State: roughly means the values of its *instance variables*.
- Behaviour: sending certain messages to an object will trigger a certain behaviour.
- Identity: objects have a “notion of self” and that two *different* objects can be exactly the same modulo their identity.

Bundling State and Behaviour

- Refers to an object containing both a set of values and a set of operations that use or modify these values.
- Most (*but not all*) Object-Oriented Programming Languages (OOPL) are class-based and use a concept of classes as the unit bundling state and behaviour together.

Bundling State and Behaviour, *Cont'd*

- A class represents a concept (what constitutes a point) whereas objects are actual instances of the concept (e.g, a point at coordinates 47,11).
- An alternative way of thinking is that instances can be *class*ified in terms of common properties, so classes as descriptions of commonalities of existing objects.

Bundling State and Behaviour, *Cont'd*

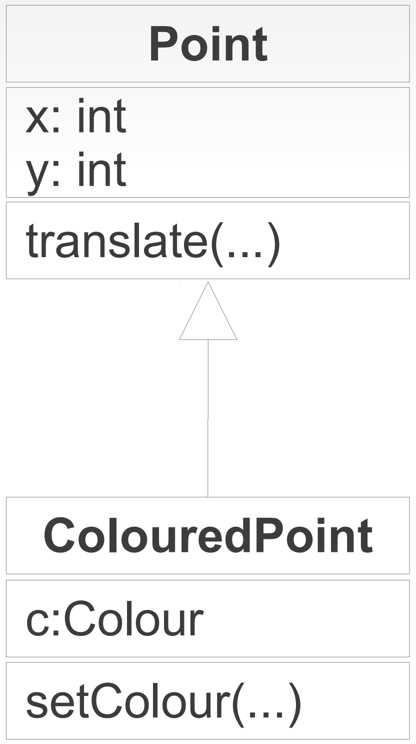
- **Question:** Relate the second way of thinking about classes to making an OO design from an existing or physical system.
- **Question:** What is the difference between a class and a Java interface?

Inheritance

- The notion of class inheritance allows defining a new class as an extension to an existing one (called *subclassing*)
- *Scandinavian school*: inheritance is understood to create a semantic or conceptual is-a relation between two classes.
- *American school:* inheritance is viewed as a code reuse mechanism.
- In American one we might see subclasses that would “make little sense in nature.”

Inheritance, *Cont'd*

- Is the following inheritance American or Scandinavian?



Inheritance, *Cont'd*

In practice in industry, more than three levels of inheritance are seldom used, and most classes do not rely on inheritance.

Polymorphism

- Refers to an object's ability to “take on many forms”.
- Meaning that a single object can have several different interfaces (sets of understood messages).
- **Subtype Polymorphism** is the most common use of polymorphism in OOP.
    - Where an instance of some class *C* can pass as an instance of any class *S* if *S* is supertype of *C*:

```
Point p = new ColouredPoint();
```

Polymorphism, *Cont'd*

- What is the difference between a class and a type?
- What is the difference between subclass and subtype?

Static and Dynamic Binding

- Binding is the tying of a name to a value or thing.
- **Static Binding:** the binding is determined at compile time, *i.e.* using the information available offline to the compiler for all programs regardless of input values, etc.
- **Dynamic Binding:** the binding is delayed to run-time and may then use any available information.
- C and C++ use static binding only.
- Java uses dynamic binding (with some exception).

Static and Dynamic Binding, *Cont'd*

- Static binding is very efficient
  - Can transfer a method call into a procedural call.
- Dynamic binding is slower, but is more powerful and flexible.

Static and Dynamic Binding, *Cont'd*

```
class Foo{
    public int value(){return 5;}
}
class Bar extends Foo{
    public int value(){return 7;}
}
Foo foo = new Bar();
System.out.println(foo.value());
```

Static and Dynamic Binding, *Cont'd*

- If static binding is used, the output will be 5
- In dynamic binding, the output will be 7
- Java uses static binding in places where the result of dynamic binding would always yield the same result. Can you think of one or more such places with respect to method calls?

Composition and Aggregation

- A key issue with objects is their granularity.
- Generally an object does not accomplish much on its own –a person is probably composed of string objects for name and social security number, integer objects for age, shoe size, etc.

Composition and Aggregation, *Cont'd*

- **Aggregation:** means that an object "is a part of" another object.
  - *A course aggregates students, the same student can be aggregated by several courses.*
- **Composition:** is a stronger notion of aggregation where if A is composed of B (etc.), then there exists no other object $C \neq A$ such that C is also composed of B.
  - *A hand is composed of fingers, no two hands share the same finger.*

Importantly

- Object-Orientation is no silver bullet: a program will not run faster, be more easily maintained, consume less resources, be developed faster, etc. because it is object-oriented.
- All such things happen because of careful thinking, design and implementation, etc. –it is due to people, neither processes nor programming paradigms.