Object Oriented Programming

COMP360

Object Oriented Programming

- Programs are defined by their data instead of by their processes (as in procedural programming)
- Programmers can create new data types as classes
- Classes contain data items and methods that can act upon the data

Classes and Objects

- A **class** defines the data and methods available
- An **object** is an instantiation of a class
- You can have many objects of the same class

Objects

- An object holds data and has methods that can act upon the data
- A class usually represents an idea
- All methods that manipulate the class are part of the class

Three Pillars of OO Programming

- Inheritance
- Encapsulation & Abstraction
- Polymorphism

Encapsulation without Objects

- Many early languages allowed you to group data into a unit
  - Cobol has records
  - C has structs
- These groupings do not provide any encapsulation or abstraction
- Methods or functions are not linked to the groupings

Methods without Objects

- If you want a function to act upon a collection of data, you need to pass the data as a parameter
- Without polymorphism, you need a separate function for each type of parameter

```
int       abs(int number)
long      labs(long number)
double fabs(double number)
float     fabsf(float number)
```

- In C, a void pointer (pointer with no type) provides a primitive way to support polymorphism

Simula

- The programming concept of objects was introduced in the mid-1960s with Simula 67, a programming language designed for simulation, created by Ole-Johan Dahl and Kristen Nygaard of the Norwegian Computing Center
- Used automatic garbage collection
- Simula was created to model physical objects. Objects had data and methods to do things with the model

information from Wikipedia

Smalltalk

- Smalltalk was an early object-oriented language
- It has all three of the characteristics of OO
- It's based on the thesis work of Alan Kay at Utah in the late 1960‘s
- It went through 5 generations at Xerox PARC, where Kay worked after graduating
- Strongly tied to the IDE

information from Wikipedia

OOP Mainstream

- Object-oriented programming developed as the dominant programming methodology in the early and mid 1990s when programming languages supporting the techniques became widely available
- Objects were added to existing languages

information from Wikipedia

Value and Reference objects

- A value object always has a value
- Value objects are instantiated when they are defined
- Reference objects only have a value when an object is created
- C++ has value and reference objects while Java has only reference objects

Java Example

```
Widget goat = new Widget(3);
Widget cow = new Widget(4);
cow = goat;                 //  reference copy
goat.setValue(7);
goat.printValue();          // displays 7
cow.printValue();           // displays 7
```

C++ Example

```
Widget goat = new Widget(3);
Widget cow = new Widget(4);
cow = goat;                 // object copy
goat.setValue(7);
goat.printValue();          // displays 7
cow.printValue();           // displays 3
```

Instance and Class Variables

- An **instance** variable exists in every object
- Memory is allocated for instance variables for every object created
- **Class** variables only exist once per class
- Memory is only allocated for a class variable once regardless of how many objects are created
- In C++ and Java class variables are defined using the keyword **static**

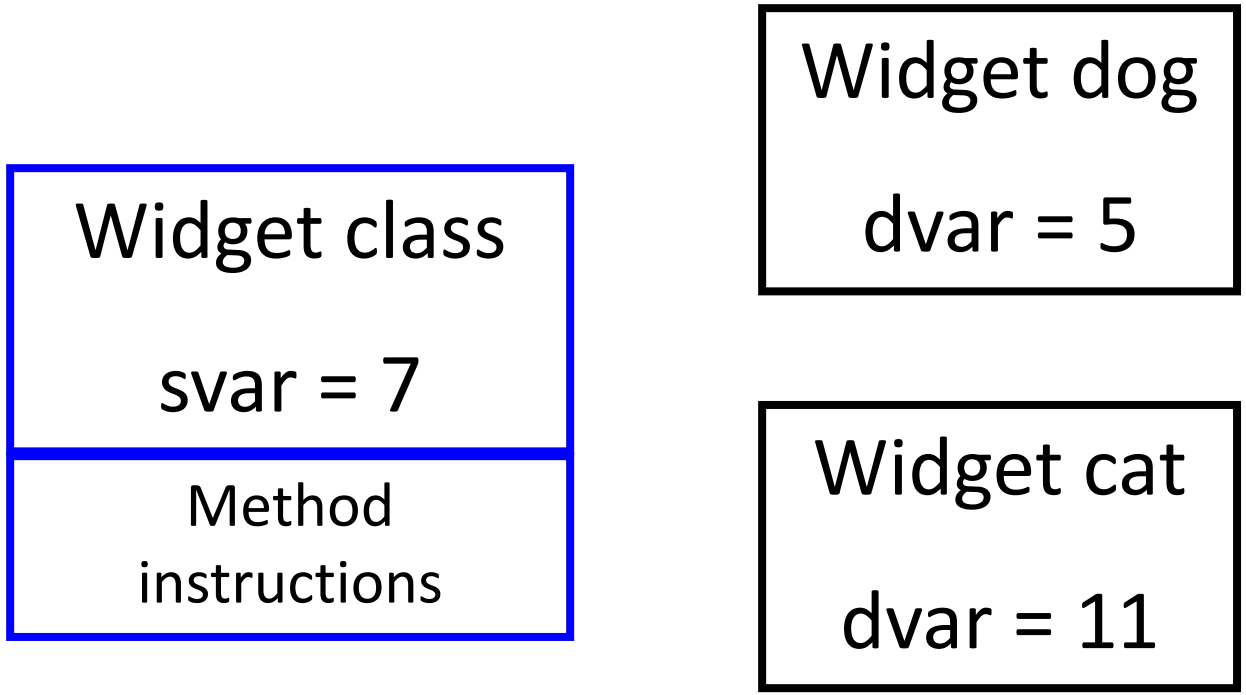
Static difference

```
class Widget {
 public:
 static int    svar;   // class variable
        int    dvar;   // object variable
}
Widget dog, cat;
dog.svar = 3;   dog.dvar = 5;
cat.svar = 7;   cat.dvar = 11;
```

- printing dog and cat shows

dog.svar is 7, cat.svar is 7 dog.dvar is 5, cat.dvar is 11

Allocation of Object



Class Loader

- Java and C# dynamically load a class when an object of that type is instantiated
- When you start execution of any Java program, over a thousand classes are loaded
- The Java class loader is a Java class, **java.lang.ClassLoader**

Inheritance

- Classes can be extended or inherited to make a new subclass
- The subclass inherits the data items and methods of all parent classes
- An object of a subclass is also an object of all parent classes
- Classes can inherit from other classes which inherit from other classes, etc.

Example Inheritance List

java.lang.Object
 |
 +--java.awt.Component
      |
      +--java.awt.Container
           |
           +--javax.swing.JComponent
                |
                +--javax.swing.AbstractButton
                     |
                     +--**javax.swing.JButton**

Polymorphism

- Polymorphism is the ability for an operation's action to depend upon the data type being used
- You can define multiple methods with the same name, but different parameter types
- The appropriate method will be called based on the parameter type

Polymorphism example

```
double doit(Triangle x) { … }
double doit(Square x) { … }

Triangle t;
Square  s;

result = doit(s) + doit(t);
```

Encapsulation

- A language mechanism for restricting direct access to some of the object's components
- There are variables that are necessary for the program to run and a subset which need to be visible outside of the object
- Programmers who use an object are restricted to access it by the provided methods or public values

Restricted Access

- Private clause for hidden entities
- Public clause for interface entities
- Protected clause for inheritance

Abstraction

- Data abstraction and encapsulation are closely related
- Abstraction makes it easier to think about a concept - hides what doesn't matter to programmer
- Protection - prevent access to things you need not see
- Plug compatibility
- Replacement of pieces, often without recompilation, definitely without rewriting libraries
- Division of labor in software projects