Polymorphism

From Chapter 5

Polymorphism

Required features

- Two or more classes related through inheritance (i.e., generalization)
  - Inheritance hierarchy may be tall, wide, or both
  - Polymorphism takes place between directly related classes
- Method overriding
  - Polymorphism is the dynamic (i.e., at runtime) selection of the correct overridden method
- Object upcasting
  - Correct method is selected from a class in the upcast path

Relative Addressing (”Real World”)
Locating items by address and offset

Relative Addressing (System Memory)
Locating features with address arithmetic

Implementing Inheritance
Symbol table and memory views

Finding Features and Up-casting
Searching the symbol table
Safe and Unsafe Casting

Converting object types

Polymorphism

Many shapes

- Non-polymorphic methods are “bound”
  - at compile time
  - called early binding or static binding
- Polymorphic methods are “bound”
  - at run time
  - called late binding or dynamic binding (also called dynamic dispatch)
- Alternate views of polymorphism:
  - One objects sends a message to another object without caring the type of the receiving object
  - The receiving object responds to a message appropriately for its type
- Java™ methods are polymorphic by default
- Third defining feature of the object-oriented model

Implementing Polymorphism

inside the JVM

Polymorphism Example

Dynamic binding

Java™ Program Example

Simple polymorphism example

class Poly // polymorphism example
{
    public static void main(String[] args)
    {
        Person[] people = new Person[4]; // instantiate array
        people[0] = new Person(); // instantiate objects; fill array
        people[1] = new Employee(); // up-cast
        people[2] = new Engineer(); // up-cast
        people[3] = new Manager(); // up-cast
        for (int i = 0; i < 4; i++)
            people[i].toString(); // polymorphic call to toString
    }
} // class Poly

Determining Which Method Is Called

Polymorphism vs non-polymorphism

- Person S = new Employee();
- void render(Person S);
  - render(new Person());
  - render(new Employee());
- Non-polymorphic call (default in C++)
  - Method/function belongs to the class named on the left hand side of the assignment operator
- Polymorphic call (default in Java™)
  - Method/function belongs to the instantiated class on the right hand side of the assignment operator
- Example
  - Non-polymorphic: S.calcPay(); // Person calcPay method
  - Polymorphic: S.calcPay(); // Employee calcPay method