Polymorphic Sorting

What is Polymorphism?

- *Polymorphism* means having many forms.
- In biology: jaguar and leopard.
- In Python: operator overloading, e.g., +
- In Java: operator overloading, e.g., +
- In OOP: ability of a method to handle different types of objects.
- Example: one sort method for integers, doubles, Strings, Objects

Built-in Python Examples

Python is a *weakly typed language*. So polymorphism is *implicit* throughout, even here:

```
x = 'hello'      # x's type is string
print x          # print a string
x = 500          # x's type is int
print x          # print a number
```


Python Selection Sort

This selection sort function sorts any type of Python type.

```
def sort(list):
    for i in range(len(list)):
        minIndex = i
        for j in range(i + 1, len(list)):
            if list[j] < list[minIndex]:
                (list[j], list[minIndex]) = (list[minIndex], list[j])

numbers = [10, 100, 5, 90, 3, 25, 62, 7];
strings = ["delta", "alpha", "gamma", "tau", "beta"]
sort(numbers)
print numbers      # prints [3, 5, 7, 10, 25, 62, 90, 100]
sort(strings)
print strings      # prints ['alpha', 'beta', 'delta', 'gamma', 'tau']
```

Java Selection Sort

Only works on int type.

```
public void sort(int A[]) {
  for (int i = 0; i < A.length - 1; i++) {
      int minIndex = i;
      for (int j = i + 1; j < A.length; j++)
         if (A[j] < A[minIndex])
            minIndex = j;
      if (minIndex != i)
            swap(A[i], A[minIndex]);
   }
}
```


Python Polymorphic Comparison

The < operator is polymorphic. It is based on **__cmp__**.

A different < is used depending on list's type:

```
def sort(list):
    for i in range(len(list)):
        minIndex = i
        for j in range(i + 1, len(list)):
            if list[j] < list[minIndex]:
                (list[j], list[minIndex]) = (list[minIndex], list[j])
```


Needed: Polymorphic <

We need < to work on many different types.

```
public void sort(int A[]) {
  for (int i = 0; i < A.length - 1; i++) {
      int minIndex = i;
      for (int j = i + 1; j < A.length; j++)
         if (A[j] < A[minIndex])
            minIndex = j;
      if (minIndex != i)
            swap(A[i], A[minIndex]);
   }
}
```


Polymorphic Compare

Java has the Comparable interface -- similar to Python's __cmp__ function.

http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Comparable.html

```
// Returns <0, 0, >0 depending if this object is
// <, ==, > the other object
public interface java.lang.Comparable {
    public int compareTo(Object other):
}
```

Java Polymorphic Sort

In Java we sort Comparable Objects:

```
public void sort (Comparable arr[]) {
    for (int i = 0; i < arr.length - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < arr.length; j++)
            if (arr[j].compareTo(arr[minIndex]) < 0)
                minIndex = j;
        if (minIndex != i) {
            Comparable temp;
            temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }
}
```


Example Java Programs

- PolySort.java – contains a polymorphic sort method.
- Animal.java – defines the compareTo() method so that Animals are Comparable.

Many Java classes implement Comparable

Integer, Float, Double, String, Character, Object, … and many others.

They define **compareTo()** for their types.

```
public interface java.lang.Comparable {

    public int compareTo(Object o):
}
```


Sorting Integer, Double, String

In Java we sort Comparable Objects:

```
Integer iArr[] = new Integer[MAXSIZE];
Double dArr[] = new Double[MAXSIZE];
String sArr[] = new String[MAXSIZE];
for (int k = 0; k < MAXSIZE; k++) {
  iArr[k] = new Integer((int)(Math.random() * 1000));  // Random ints
  dArr[k] = new Double((Math.random() * 1000));     // Random doubles
  // Random Strings
  sArr[k] = new String(""+ (char)('a' + (int)(Math.random()* 26)));
}

sort(iArr);
sort(dArr);
sort(sArr);
```