Recap

- We can sort things using insertion sort!
- We can sort strings of characters
- We can sort lists of integers
- But what happens if, say, we want to sort a list of people by first name?
- Or last name?
- Or age?
- Do we have to write the insertion sort code all over again each time? Fortunately not.

Recap: string insertion sort

```
1   String insert(char c, String s) {
2     String to_return = "";
3     while(true) {
4       if(s.equals("")) return to_return+c;
5       char c2 = s.charAt(0);
6       if(c>c2) { to_return = to_return + c2; }
7       else { return to_return+c+s; }
8       s=s.substring(1);
9     }
10  }
```

Recap: interfaces

What are they again?

. . .

```
1   String ins_sort(String s) {
2     String to_return = "";
3     while(true) {
4       if(s.equals("")) return to_return;
5       char c = s.charAt(0);
6       to_return = insert(c,to_return);
7       s=s.substring(1);
8     }
9   }
```

Recap: integer list insertion sort

```
1    List insert(Integer i, List l0) {
2      List to_return = nil();
3      while(true) {
4        if(l0.isEmpty()) return append1(to_return,i);
5        Integer i2 = (Integer)(hd(l0)); // note cast!!!
6        if(i>i2) { to_return = append1(to_return,i2); }
7        else { return append(append1(to_return,i),l0); }
8        l0=tl(l0);
9      }
10   }
```

(And similar for `ins_sort`)

. . .

```
1      Object insert(Object o, Object l) {
2        Object to_return = ops.u_nil();
3        while(true) {
4          if(ops.u_is_empty(l)) {
5            return ops.u_append1(to_return,o);}
6          Object o2 = ops.u_hd(l);
7          if(ops.u_lt(o2,o)) {
8            to_return = ops.u_append1(to_return,o2); }
9          else {
10           return ops.u_append(
11             ops.u_append1(to_return,o),l); }
12         l = ops.u_tl(l);
13       }
14     }
```

Problem: it is too easy to get the element type and the list type confused - everything is an object.

Generalizing: insertion sort for strings and int lists

We need code that is:

- GENERIC - works for strings, lists of ints, lists of string, lists of persons, etc.

```
1    static interface Ops {
2      // u_ stands for "untyped"
3      boolean u_lt(Object o1, Object o2);
4      Object u_nil();
5      Object u_cons(Object x, Object l);
6      Object u_append(Object l1, Object l2);
7      Object u_append1(Object l, Object e);
8      boolean u_is_empty(Object l);
9      Object u_hd(Object l);
10     Object u_tl(Object l);
11   }
```

(See file `Untyped_generic_isort.java`)

Improving this code by adding more informative types

We need code that is:

- GENERIC - works for strings, lists of ints, lists of string, lists of persons, etc.
- TYPED - types that distinguish between the "list like" type (string, list etc) and the "element" type (char, int etc.).

First some syntax: List<Integer>

- Till now we have been using plain Java `List`.
- This is fine; but all it tells you about the elements of the list is that they are `Objects`.
- Early Java supported only this syntax. But it was known for a long time (30 years or more) that this was not sufficient.
- What is needed is for the "list" type to somehow tell you what the type of the elements is.
- For this reason, the syntax `List<Integer>` (or more generally `List<X>` for some type `X`) was introduced. This type means "list of integers".
- The type `List<X>` is a "generic type" which can be "instantiated" to give a particular type such as `List<Integer>`.
- When you take the head of a `List<Integer>`, Java knows that that the object is an `Integer`. You no longer have to cast it to a particular desired type `Integer i = (Integer)(hd(xs))`.

Example

```
List<Integer> xs = cons(1,cons(2,cons(3,nil())));
int x = hd(xs); // no casting
// etc
```

To make this actually work, you need to add some types to `hd`:

```
E hd(List<E> xs) {  // used to be: Object hd(List xs)
  ...
}
```

Typed generic insertion sort (TGIS)

Now we can implement TYPED GENERIC insertion sort.

- Handles strings, lists of ints, lists of strings etc.
- Types make sure we don't confuse lists with elements of lists etc.

TGIS: comparing two elements

In the following, we have an interface which is "parametric" on the type of elements; we have called the type of element `E`, but later we will "instantiate" the type `E` to be `Character`, `Integer` etc.

```
interface Comp<E> {
  public boolean c_lt(E e1, E e2);
}
```

TGIS: "list like" operations

```
interface Ops<E,L> {
  L g_nil();
  L g_cons(E x, L l);
  L g_append(L l1, L l2);
  L g_append1(L l, E e);

  boolean g_is_empty(L l);
  E g_hd(L l);
  L g_tl(L l);
}
```

TGIS: insert

```
  L insert(E o, L l) {
    L to_return = ops.g_nil();
    while(true) {
      if(ops.g_is_empty(l)) {
        return ops.g_append1(to_return,o); }
      E o2 = ops.g_hd(l);
      if(comp.c_lt(o2,o))
        to_return = ops.g_append1(to_return,o2);
      else {
        return ops.g_append(
          ops.g_append1(to_return,o),l); }
      l = ops.g_tl(l);
    }
  }
```

TGIS: isort

```
  L isort(L l) {
    L to_return = ops.g_nil();
    while(true) {
      if(ops.g_is_empty(l)) return to_return;
      E o = ops.g_hd(l);
      to_return = insert(o,to_return);
      l = ops.g_tl(l);
    }
  }
```

TGIS: insert and isort in a class

```
class Sorter<E,L> {
  public Sorter(Comp<E> c, Ops<E,L> o)...

  L insert(E o, L l) ...

  L isort(L l) ...
}
```

- Note: element type `E` and list type `L`

Using TGIS: sort a string

```
1       Sorter<Character,String> x =
2         new Sorter(
3                    new C_comp(),
4                    new String_ops()
5                    );
6       System.out.println(x.isort("defabcg"));
```

- What are `E` and `L` here?
- How do we compare characters?
- How do we get the "head" of a string? The empty string?

Using TGIS: sort a list of integers

```
1       Sorter<Integer,List<Integer>> x =
2         new Sorter(
3                    new I_comp(),
4                    new I_list_ops()
5                    );
6       List<Integer> l =
7         java.util.Arrays.asList(new Integer[] {4,1,2,0});
8       System.out.println(x.isort(l));
```

- What are `E` and `L` here?
- How do we compare ints?
- How do we get the "head" of a list of ints? The empty list of ints?

Check your understanding

- Given the above, how would you use TGIS to sort a list of persons?
- By first name?
- By last name?
- Write some code to sort some lists of persons.