CSE214 Data Structures
Generics

YoungMin Kwon

Generics

- Class or a method that work for different types of objects
  - E.g. A sorting method that works for Integers and Strings
- Type variables can be used to work with various types
  - The real type for the type variables are set when the generic class or the generic method is used

```
public class Generic {
    public static interface Ordered {
        public boolean ge(Ordered a);   //greater than or equal to

        //default methods
        public default boolean gt(Ordered a) {      //greater than
            return this.ge(a) && this.ne(a);
        }
        public default boolean le(Ordered a) {      //less than or equal to
            return a.ge(this);
        }
        public default boolean lt(Ordered a) {      //less than
            return !this.ge(a);
        }
        public default boolean eq(Ordered a) {      //equal
            return this.ge(a) && a.ge(this);
        }
        public default boolean ne(Ordered a) {      //not equal
            return !this.eq(a);
        }
    }
```

```
public static class Int implements Ordered {
    public int n;
    public Int(int n)            { this.n = n; }
    public String toString()     { return Integer.toString(n); }
    public boolean ge(Ordered a) {
        Int that = (Int)a;
        return this.n >= that.n;
    }
}

public static class Str implements Ordered {
    public String str;
    public Str(String str)       { this.str = str; }
    public String toString()     { return str; }
    public boolean ge(Ordered a) {
        Str that = (Str)a;
        return this.str.compareTo(that.str) >= 0;
    }
}
```

```java
//Find the index of the minimum element in arr[from...]
public static <E extends Ordered> int findMinIndex(E[] arr, int from) {
    E min = arr[from];    //minimum element
    int minIndex = from;  //index of min
    //TODO: find the index of the minimum element of arr[from...]

    return minIndex;
}

//Selection sort
public static class SelSort<E extends Ordered> {
    public void sort(E[] arr) {
        int n = arr.length;

        for(int i = 0; i < n; i++) {
            //TODO: find the min index j in arr[i..n] and
            //      swap arr[i] and arr[j]
        }
    }
}

public static <E> void swap(E[] arr, int i, int j) {
    E t = arr[i]; arr[i] = arr[j]; arr[j] = t;
}
```

```java
//Revert the array: [1, 2, 3] -> [3, 2, 1]
public static void revert(Object[] arr) {
    int n = arr.length;

    //TODO: swap arr[0] and arr[n-1], swap arr[1] and arr[n-2], ...
}

public static void print(Object[] arr) {
    for(int i = 0; i < arr.length; i++)
        System.out.print(" " + arr[i]);
    System.out.println("");
}
```

```
    public static void main(String[] args) {
        Int[] iarr = new Int[] { new Int(5), new Int(3), new Int(7),
                                 new Int(1), new Int(4), new Int(2) };
        Str[] sarr = new Str[] { new Str("s"), new Str("a"), new Str("m"),
                                 new Str("p"), new Str("l"), new Str("e") };


        //test findMinIndex
        System.out.println("min indx: " + findMinIndex(iarr, 0));
        System.out.println("min indx: " + findMinIndex(sarr, 0));

        //test sort
        new SelSort<Int>().sort(iarr);
        print(iarr);
        new SelSort<Str>().sort(sarr);
        print(sarr);

        //test revert
        revert(iarr);
        print(iarr);
        revert(sarr);
        print(sarr);
    }
}
```

Expected output

```
min indx: 3
min indx: 1
 1 2 3 4 5 7
 a e l m p s
 7 5 4 3 2 1
 s p m l e a
```