src\main\java\shelf\Shelf.java

```java
package shelf;

import java.util.Comparator;
import java.util.Iterator;
import java.util.NoSuchElementException;

public class Shelf<T extends ShelfItem> implements Iterable<T>{

  private T upperLeft;
  private T upperRight;
  private T lowerLeft;
  private T lowerRight;

  public T get(int index){
    switch (index){
      case 0: return upperLeft;
      case 1: return upperRight;
      case 2: return lowerLeft;
      case 3: return lowerRight;

      default: throw new IllegalArgumentException("invalid index");
    }
  }

  public void set(int index, T value){
    switch (index){
      case 0: upperLeft = value; break;
      case 1: upperRight = value; break;
      case 2: lowerLeft = value; break;
      case 3: lowerRight = value; break;

      default: throw new IllegalArgumentException("invalid index");
    }
  }

  @Override
  public Iterator<T> iterator() {
    return new ShelfIterator();
  }

  private class ShelfIterator implements Iterator<T> {

    private int next = 0;

    @Override
    public boolean hasNext() {
      return (next < 4);
    }

    @Override
    public T next() {

      if (!hasNext())
        throw new NoSuchElementException("...");

      return Shelf.this.get(next++);

    }

  }

  public void takeFrom(Shelf<? extends T> other){
    if (other == null)
      throw new IllegalArgumentException("other must not be null");

    for (int i = 0; i < 4; i++) {
      this.set(i, other.get(i));
      other.set(i, null);
    }
  }

  public void swap(Shelf<T> other){

    for (int i = 0; i < 4; i++){
      var temp = this.get(i);
      this.set(i, other.get(i));
      other.set(i, temp);
    }

  }

  public T max(Comparator<T> comparator){

    if (comparator == null)
      throw new IllegalArgumentException("comparator must not be null");

     T maxItem = null;

     for (int i = 0; i < 4; i++){
       if (get(i) != null && (maxItem == null || comparator.compare(maxItem, get(i)) < 0))
{
         maxItem = get(i);
       }

    }

    return maxItem;

  }

  public static <S extends ShelfItem> void transferAndTrim(Shelf<? extends S> from, Shelf<
? super S> to){

    if (from == null)
      throw new IllegalArgumentException("from must not be null");

    if (to == null)
      throw new IllegalArgumentException("to must not be null");

    int toIndex = 0;

    for (int i = 0; i < 4; i++){
      to.set(i, null);
    }

    for (int i = 0; i < 4; i++){

      if (from.get(i) != null){

        to.set(toIndex, from.get(i));

        from.set(i, null);

        toIndex++;


      }


    }



  }



  public T getUpperLeft() {
    return upperLeft;
  }

  public void setUpperLeft(T upperLeft) {
    this.upperLeft = upperLeft;
  }

  public T getUpperRight() {
    return upperRight;
  }

  public void setUpperRight(T upperRight) {
    this.upperRight = upperRight;
  }

  public T getLowerLeft() {
    return lowerLeft;
  }

  public void setLowerLeft(T lowerLeft) {
    this.lowerLeft = lowerLeft;
  }

  public T getLowerRight() {
    return lowerRight;
  }

  public void setLowerRight(T lowerRight) {
    this.lowerRight = lowerRight;
  }

}

```