Merge and Quick Sorts

Code from Textbook

```
public static <AnyType extends Comparable<? super
AnyType>>
    void mergeSort( AnyType [ ] a )
    {
        AnyType [ ] tmpArray = (AnyType[])
                                    new Comparable[ a.length ];
        mergeSort( a, tmpArray, 0, a.length - 1 );
    }
```

```
private static <AnyType extends Comparable<? super
      AnyType>> void mergeSort( AnyType [ ] a, AnyType [ ]
                              tmpArray, int left, int right )
   {
      if( left < right )
      {
         int center = ( left + right ) / 2;
         mergeSort( a, tmpArray, left, center );
         mergeSort( a, tmpArray, center + 1, right );
         merge( a, tmpArray, left, center + 1, right );
      }
   }
```

```
private static <AnyType extends Comparable<? super
                AnyType>> void merge( AnyType [ ] a,
  AnyType [ ] tmpArray, int leftPos, int rightPos, int rightEnd )

   {
     int leftEnd = rightPos - 1;
     int tmpPos = leftPos;
     int numElements = rightEnd - leftPos + 1;
```

```
while( leftPos <= leftEnd && rightPos <= rightEnd )
    if( a[ leftPos ].compareTo( a[ rightPos ] ) <= 0 )
        tmpArray[ tmpPos++ ] = a[ leftPos++ ];
    else
        tmpArray[ tmpPos++ ] = a[ rightPos++ ];

while( leftPos <= leftEnd )
    tmpArray[ tmpPos++ ] = a[ leftPos++ ];

while( rightPos <= rightEnd )
    tmpArray[ tmpPos++ ] = a[ rightPos++ ];
    for( int i = 0; i < numElements; i++, rightEnd-- )
        a[ rightEnd ] = tmpArray[ rightEnd ];
}
```

Quicksort

```
public static <AnyType extends Comparable<? super
            AnyType>> void quicksort( AnyType [ ] a )
   {
       quicksort( a, 0, a.length - 1 );
   }
```

```
private static final int CUTOFF = 3;

public static <AnyType> void swapReferences(
        AnyType [ ] a,  int index1, int index2 )
{
   AnyType tmp = a[ index1 ];
   a[ index1 ] = a[ index2 ];
   a[ index2 ] = tmp;
}
```

```
private static <AnyType extends Comparable<? super
      AnyType>> void quicksort( AnyType [ ] a, int left, int right )
   {
      if( left + CUTOFF <= right )
      {
         Recursive Case -- Heart of Quicksort
      }
      else
         insertionSort( a, left, right );
   }
```

```
private static <AnyType extends Comparable<? super
AnyType>> AnyType median3( AnyType [ ] a, int left, int right )
    {
        int center = ( left + right ) / 2;
        if( a[ center ].compareTo( a[ left ] ) < 0 )
            swapReferences( a, left, center );
        if( a[ right ].compareTo( a[ left ] ) < 0 )
            swapReferences( a, left, right );
        if( a[ right ].compareTo( a[ center ] ) < 0 )
            swapReferences( a, center, right );

        swapReferences( a, center, right - 1 );
        return a[ right - 1 ];
    }
```

Heart of Quicksort

```
AnyType pivot = median3( a, left, right );
int i = left, j = right - 1;
for( ; ; )
{
    while( a[ ++i ].compareTo( pivot ) < 0 ) { }
    while( a[ --j ].compareTo( pivot ) > 0 ) { }
    if( i < j )
        swapReferences( a, i, j );
    else
        break;
}
```

```
swapReferences( a, i, right - 1 );

quicksort( a, left, i - 1 );

quicksort( a, i + 1, right );
```