CSE 332 - Section 1 Worksheet

1. TypeSubtypePair

The first item is of some type, the second item is of a subtype of the first.

```
public class TypeSubtypePair <_____________________________>{
    private A first;
    private B second;
    public TypeSubtypePair(A first, B second){
        this.first = first;
        this.second = second;
    }
    public A getFirst(){
        return first;
    }
    public B getSecond(){
        return second;
    }
    public void setFirst(A newFirst){
        this.first = newFirst;
    }
    public void setSecond(B newSecond){
        this.second = newSecond;
    }
    public String toString(){
        return "(" + first + "," + second+ ")";
    }
}
```

2. IntOtherPair

The first item is an integer, the second item is some other type.

```
public class IntOtherPair<_____________________>{
    private ________ first;
    private ________ second;
    public IntOtherPair(________ first, ________ second){
        this.first = first;
        this.second = second;
    }
    public ________ getFirst(){
        return first;
    }
    public ________ getSecond(){
        return second;
    }
    public void setFirst(________ newFirst){
        this.first = newFirst;
    }
    public void setSecond(________ newSecond){
        this.second = newSecond;
    }
    public String toString(){
        return "(" + first + "," + second+ ")";
    }
}
```

3. ItemPairPair

The first item is of some type, the second item is a LikePair of things whose type inherits the first item's.

```
public class ItemPairPair<_____________________________>{
    private ________ first;
    private ________ second;
    public ItemPairPair(________ first, ________ second){
        this.first = first;
        this.second = second;
    }
    public ________ getFirst(){
        return first;
    }
    public ________ getSecond(){
        return second;
    }
    public void setFirst(________ newFirst){
        this.first = newFirst;
    }
    public void setSecond(________ newSecond){
        this.second = newSecond;
    }
    public String toString(){
        return "(" + first + "," + second+ ")";
    }
}
```

4. ItemArrayPair

The first item is of some type, the second item is an array of objects of another type.

```
public class ItemArrayPair<_____________________>{
    private ________ first;
    private ________ second;
    public ItemArrayPair(________ first, ________ second){
        this.first = first;
        this.second = second;
    }
    public ________ getFirst(){
        return first;
    }
    public ________ getSecond(){
        return second;
    }
    public void setFirst(________ newFirst){
        this.first = newFirst;
    }
    public void setSecond(________ newSecond){
        this.second = newSecond;
    }
    public String toString(){
        return "(" + first + "," + second+ ")";
    }
}
```

5. ComparablePair

A pair of items such that both of them implement the comparable interface. A comparable pair must itself be comparable!

```
public class ComparablePair<_____________________> implements
Comparable <_____________________>{
    private ________ first;
    private ________ second;
    public ComparablePair(________ first, ________ second){
        this.first = first;
        this.second = second;
    }
    public ________ getFirst(){
        return first;
    }
    public ________ getSecond(){
        return second;
    }
    public void setFirst(________ newFirst){
        this.first = newFirst;
    }
    public void setSecond(________ newSecond){
        this.second = newSecond;
    }
    public int compareTo(__________________ other){
        int compare1st = this.first.compareTo(other.first);
        return compare1st==0 ? compare1st : this.second.compareTo(other.second);
    }
    public String toString(){
        return "(" + first + "," + second+ ")";
    }
}
```