# CSE 332 - Section 1 Worksheet Solution

## 1. TypeSubtypePair

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

```
public class TypeSubtypePair <A, B extends A>{
    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<T>{
    private int first;
    private T second;
    public IntOtherPair(int first, T second){
        this.first = first;
        this.second = second;
    }
    public int getFirst(){
        return first;
    }
    public T getSecond(){
        return second;
    }
    public void setFirst(int newFirst){
        this.first = newFirst;
    }
    public void setSecond(T 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 is a subclass of the first item's type.

```
public class ItemPairPair<A, B extends A>{
    private A first;
    private LikePair<B> second;
    public ItemPairPair(A first, LikePair<B> second){
        this.first = first;
        this.second = second;
    }
    public A getFirst(){
        return first;
    }
    public LikePair<B> getSecond(){
        return second;
    }
    public void setFirst(A newFirst){
        this.first = newFirst;
    }
    public void setSecond(LikePair<B> 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<A,B>{
    private A first;
    private B[] second;
    public ItemArrayPair(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+ ")";
    }
}
```

## 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<A extends Comparable<A>, B extends Comparable<B>>
implements Comparable<ComparablePair<A,B>> {
// This one's a doozy! Here's what it means: We have the class named
ComparablePair. It is parameterized by 2 types, A and B. Both of those types
inherit from Comparable, and the type of thing they are able to compare to is
themselves (which is why we see "A extends Comparable<A>"). The ComparablePair
class d also implements the Comparable interface but here we also need to state
what it can be compared to, which is another instance of itself.
    private A first;
    private B second;
    public ComparablePair(A first, B second){
        this.first = first;
        this.second = second;
    }
    public int compareTo(ComparablePair<A,B> other){
        int compare1st = this.first.compareTo(other.first);
        return compare1st==0 ? compare1st : this.second.compareTo(other.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+ ")";
    }
}
```