Andrew Martin
Charles Ceccardi
Mitchell Oneka
Samuel Mitchell

3.1 First-Cut:ObjectElements

2. Answer the following question in a word processor file:
Why does this line of code compile?
stadiums.put("Bridgeforth Stadium", 25000)

Stadium connects to the ObjectMap class and the ObjectMap class has a method called “put” that takes two arguments.

3.2 Revision: Generic Elements

4. Answer the following questions in a word processor file:

(a) What happens if you re-introduce the problem that you fixed in Section 3.1, Step 1?
You need to fix it again while also including the generics

(b) Will the resulting code compile?
No because main is trying to access a non static thing

(c) Why do you think generic collection classes are sometimes called ”type-safe” collections?
It is something with a flexible type.

5. Answer the following questions in a word processor file:

(a) List some reasons that the Generic Twople class might be preferable to the Object Twople class.
It simplifies the process by making the generic and gives them peramiterized types

(b) Can you think of any situations where the Object Twople class might be preferable
If you need things without parameters or working in a setting where multible people are going to be seeing the code.

3.3 Wildcards and Subclasses
Copy the contents of CompilerIssues.java (Listing 5) into a word processor file:
Week11LabReport.doc.
Comment each assignment statement with either:
// Compiles
Or
// Does Not Compile
For those lines that will not compile, include an explanation of the problem.
Once you have finished all of the statements, check your answers by attempting to compile.

```
public class CompilerIssues {
	public static void main(String[] args) {

		GenericTwople<String, Integer> p1; // Compiles
		GenericTwople<String, Number> p2; // Compiles
		GenericTwople<Object, Object> p3;// Compiles
		GenericTwople<?, ?> p4;// Compiles
		GenericTwople<?, ? extends Number> p5;// Compiles

		p1 = new GenericTwople<String, Integer>("A", 7);// Compiles
		int a = p1.getSecond();// Compiles

		p1 = new GenericTwople<Integer, Double>(23, 12.0); // Does Not
	Compile
		p1 was made with String and Integer and cannot be converted
	from Integer and Double.
```

```
        p2 = new GenericTwople<String, Integer>("B", 8);// Does Not
    Compile
        p2 was made with String and Number and cannot be converted
    from String and Integer.

        p3 = new GenericTwople<String, Integer>("C", 9);// Does Not
    Compile
        p3 was made with Object and Object and cannot be converted
    from String and Integer.


        p4 = new GenericTwople<String, String>("House", "Car");//
    Compiles
        p4 = new GenericTwople<String, Integer>("D", 10);// Compiles

        Integer b = p4.getSecond();// Does Not Compile
        p4 is not an Integer and cannot become Integer b.

        Integer c = (Integer) p4.getSecond();// Compiles

        p5 = new GenericTwople<String, String>("E", "G");// Compiles X
        p5 = new GenericTwople<String, Double>("E", 11.0);// Compiles
        p5 = new GenericTwople<String, Integer>("E", 11);// Compiles

        Integer d = p4.getSecond();// Does Not Compile
        p4 is not an Integer and cannot become Integer d.

        Integer e = (Integer) p4.getSecond();// Compiles
    }
}
```