Assigned: Tuesday 16 February 2009
Due: Tuesday 23 February 2009
Topics: Parameterized types, abstract classes, inheritance, generics, collections
Collaboration: This homework assignment is to be completed individually.
Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Be sure to generate unit tests and a statement justifying their sufficiency.

This assignment may involve some subtlety with Java typing. The programming is not difficult, but it will require careful thought about generics. Thus, start early and ask lots of questions!

In mathematics, a ring is a type of algebraic structure that has a set of elements, all of the same type, and two operations, loosely called addition and multiplication. Such a pairing of a set (e.g., all integers) with these two operations is a ring when the “addition” operation is commutative and the “multiplication” operation is distributive over addition.

In this assignment, you will create a generic ring abstract class that supports these operations, and two fully concrete implementations of some familiar rings: integers with common addition and multiplication, and booleans (which feature the logical “xor” and logical “and” as its operations).

Ring Class
Define an abstract, typed class Ring<AnyType> that specifies the add and multiply operations. These should return new objects of the generic type (rather than being mutators). Note that we do not yet have any shared implementations, so this class will not actually “do” much—yet.

In addition, add two methods for the additiveIdentity and the multiplicativeIdentity. By definition,

```
    member.add( additiveIdentity() ).equals( member )
```

should always be true, and

```
    member.multiply( multiplicativeIdentity() ).equals( member )
```

should also be true.

RInteger Class
Create a (concrete) subclass of Ring called RInteger. (We can’t use just Integer because Java already has a class by that name.) What should the parameterized type be for the extends clause? If you’re confused, you might refer to the Comparable<T> interface and also see the declaration of the Integer class in the Java API. Note that Integer is implementing a generic interface, while RInteger will be extending a generic class.

Implement all the required methods of the class using the typical add and multiply operations on integers. You should also implement equals and toString methods for later testing.
RBoolean Class
Create another (concrete) subclass of Ring called RBoolean. The parameterized type will be analogous to RInteger. For boolean rings, the multiplication operation is the logical “and” (conjunction), while the addition operation is the logical “xor” (exclusive “or”/symmetric difference). You should also implement equals and toString methods for later testing.

Expanding AbstractRing
Add two static methods to your Ring class: one (called sum) that will compute the sum of all the elements in a Collection (the parent interface of ArrayList) of any objects that implement the Ring interface, and another (called product) that will compute the product of the elements in such an ArrayList. You should use Java’s enhanced for loop. We will take as a precondition that the collection is non-empty.
Every item in the given Collection must be the same kind of Ring object, otherwise you probably would not be able to add them together (i.e., what’s 5 plus true?). Thus, you will need to use type bounds on these static methods (c.f. Weiss 4.7.3 and 4.7.4). Note that Collection will need to be imported from java.util.

Testing
Demonstrate the functionality of your concrete classes and the additional static methods with a suite of appropriate unit tests that show the generic static methods working for both types of rings.