An issue with generics and arrays

This little note discusses a problem in creating an array of some class-type, say `Entry`, that uses a type parameter. The seemingly correct new-expression **`new`** `Entry[5]` doesn't work. We show how to create the array.

```
public class TimeSet<E> {
   private Entry[] s;

    /** Constructor: an empty set with
      * max size s. */
   public TimeSet(int max) {
        s= new Entry[max];
        …
    }
    …

   private class Entry {
       private E val;
       …
   }
}
```

We were writing class `TimeSet<E>`, as shown to the right. The statement given in red,

```
s= new Entry[max];
```

looks right, but the Java compiler wouldn't compile it, giving this message:

Cannot create a generic array of TimeSet<E>.Entry

Note that class `TimeSet<E>` includes inner class `Entry`, which has a field whose type is type parameter `E`, and this caused the problem.

No Java text that we had discussed this problem, and it took some time, searching stackoverflow.com and other websites, to find a solution. So don't feel bad if you also had trouble figuring out what was wrong.

Also, you probably won't remember the solution, and you don't have to. Just visit this page when the problem arises.

The solution

Replace the assignment to `s` with this statement:

```
s= (Entry[]) Array.newInstance(Entry.class, max);
```

This statement uses the reflection parts of Java, and you will have to put this import statement at the top of the file:

```
import java.lang.reflect.Array;
```