4.1.10 compatibility of array types

One of the difficulties in language design is how to handle inheritance for aggregate types. In our example, we know that `Employee` *IS-A* `Person`. But is it true that `Employee[]` *IS-A* `Person[]`? In other words, if a routine is written to accept `Person[]` as a parameter, can we pass an `Employee[]` as an argument?

Arrays of subclasses are type-compatible with arrays of superclasses. This is known as *covariant arrays*.

At first glance, this seems like a no-brainer, and `Employee[]` should be type-compatible with `Person[]`. However, this issue is trickier than it seems. Suppose that in addition to `Employee`, `Student` *IS-A* `Person`. Suppose the `Employee[]` is type-compatible with `Person[]`. Then consider this sequence of assignments:

```
Person[] arr = new Employee[ 5 ]; // compiles: arrays are compatible
arr[ 0 ] = new Student( ... );    // compiles: Student IS-A Person
```

If an incompatible type is inserted into the array, the Virtual Machine will throw an `ArrayStoreException`.

Both assignments compile, yet `arr[0]` is actually a referencing an `Employee`, and `Student` *IS-NOT-A* `Employee`. Thus we have type confusion. The runtime system cannot throw a `ClassCastException` since there is no cast.

The easiest way to avoid this problem is to specify that the arrays are not type-compatible. However, in Java the arrays *are* type-compatible. This is known as a *covariant array type*. Each array keeps track of the type of object it is allowed to store. If an incompatible type is inserted into the array, the Virtual Machine will throw an `ArrayStoreException`.

4.1.11 covariant return types

In Java 5, the subclass method's return type only needs to be type-compatible with (i.e., it may be a subclass of) the superclass method's return type. This is known as a *covariant return type*.

Prior to Java 5, when a method was overridden, the subclass method was required to have the same return type as the superclass method. Java 5 relaxes this rule. In Java 5, the subclass method's return type only needs to be type-compatible with (i.e., it may be a subclass of) the superclass method's return type. This is known as a *covariant return type*. As an example, suppose class `Person` has a `makeCopy` method

```
public Person makeCopy( );
```

that returns a copy of the `Person`. Prior to Java 5, if class `Employee` overrode this method, the return type would have to be `Person`. In Java 5, the method may be overridden as

```
public Employee makeCopy( );
```