You're also free to put boundaries on the type you declare. For example, if you want to restrict the makeArrayList() method to only Number or its subtypes (Integer, Float, and so on), you would say.

public <T extends Number> void makeArrayList(T t)

public void makeList(T t) { }// legal method

public class Radio {

public <T> Radio(T t) { } // legal constructor

}

You can declare a class with a name that is the same as the type parameter placeholder:

class X { public <X> X(X x) { } }

One of the most common mistakes programmers make when creating generic classes or methods is to use a <?> in the wildcard syntax rather than a type variable <T>, <E>, and so on. This code might look right, but isn't:

public class NumberHolder<? extends Number> { }

While the question mark works when declaring a reference for a variable, it does NOT work for generic class and method declarations. This code is not legal:

public class NumberHolder<?> { ? aNum; } // NO!

But if you replace the <?> with a legal identifi er, you're good:

public class NumberHolder<T> { T aNum; } // Yes