Note: this error code is no longer emitted by the compiler. In Rust 1.3, the default object lifetime bounds are expected to change, as described in RFC 1156. You are getting a warning because the compiler thinks it is possible that this change will cause a compilation error in your code. It is possible, though unlikely, that this is a false alarm.

The heart of the change is that where `&'a Box<SomeTrait>` used to default to `&'a Box<SomeTrait+'a>`, it now defaults to `&'a Box<SomeTrait+'static>` (here, `SomeTrait` is the name of some trait type). Note that the only types which are affected are references to boxes, like `&Box<SomeTrait>` or `&[Box<SomeTrait>]`. More common types like `&SomeTrait` or `Box<SomeTrait>` are unaffected.

To silence this warning, edit your code to use an explicit bound. Most of the time, this means that you will want to change the signature of a function that you are calling. For example, if the error is reported on a call like `foo(x)`, and `foo` is defined as follows:

```
# trait SomeTrait {}
fn foo(arg: &Box<SomeTrait>) { /* ... */ }
```

You might change it to:

```
# trait SomeTrait {}
fn foo<'a>(arg: &'a Box<SomeTrait+'a>) { /* ... */ }
```

This explicitly states that you expect the trait object `SomeTrait` to contain references (with a maximum lifetime of `'a`).