4.4.2 Inheriting Fields

When one class has another class as a superclass, it inherits all of the nonstatic fields of that class. Consider an extension to the `Greeting` class to handle greetings in Russian:

```
class RussianGreeting extends Greeting
{
    String intro = "Zdravstvuite";
}
```

An instance of `RussianGreeting` has two fields, both named `intro`. An instance of `RussianGreeting` looks like Figure 4.4. The full names of the fields are different. In Java, the English-language `intro` is *hidden* behind the Russian-language version. In Oolong, they are both equally accessible:

```
.method static internationalGreetings(LRussianGreeting;)V
aload_0              ; There is a RussianGreeting in register 0
                     ; Push "Zdravstvuite":
getfield RussianGreeting/intro Ljava/lang/String;
aload_0              ; Reload the same object
                     ; Push "Hello":
getfield Greeting/intro Ljava/lang/String;
;; Rest of the method omitted
.end method
```

At the end of the code shown, there will be two objects on the stack. The bottom of the stack will contain a `reference` to the `String` "`Zdravstvuite`", and above it will be a `reference` to the `String` "`Hello`".

4.4.3 Changing Field Values

To get the value of a field, use `getfield`, which takes an object on the stack and leaves in its place the value of the field. The counterpart of `getfield` for changing

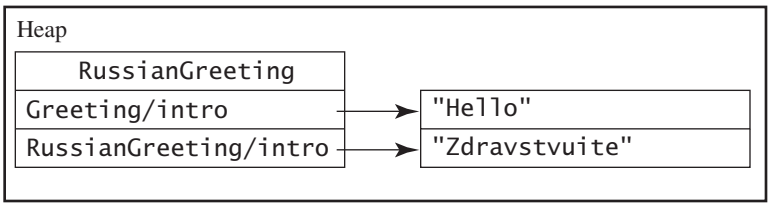

FIGURE 4.4: *An instance of* `RussianGreeting`