



## ADVANCED TOPIC 10.4

### Inheritance and the toString Method

You just saw how to write a `toString` method: Form a string consisting of the class name and the names and values of the instance fields. However, if you want your `toString` method to be usable by subclasses of your class, you need to work a bit harder. Instead of hardcoding the class name, you should call the `getClass` method to obtain a *class* object, an object of the `Class` class that describes classes and their properties. Then invoke the `getName` method to get the name of the class:

```
public String toString()
{
    return getClass().getName() + "[balance="
        + balance + "];"
}
```

Then the `toString` method prints the correct class name when you apply it to a subclass, say a `SavingsAccount`.

```
SavingsAccount momsSavings = . . . ;
System.out.println(momsSavings);
// Prints "SavingsAccount[balance=10000]"
```

Of course, in the subclass, you should override `toString` and add the values of the subclass instance fields. Note that you must call `super.toString` to get the superclass field values—the subclass can't access them directly.

```
public class SavingsAccount extends BankAccount
{
    public String toString()
    {
        return super.toString() +
            "[interestRate=" + interestRate + "];"
    }
}
```

Now a savings account is converted to a string such as `SavingsAccount[balance=10000][interestRate=5]`. The brackets show which fields belong to the superclass.