Cloning in Java

You can make a copy of any Java object using the *clone* method.

```
JObj j1 = (JObj)j0.clone();
```

The *clone* method always returns an object declared to have a return type of Object. Thus you must cast it to the actual type of the object that you are cloning. Three other significant restrictions apply to this method:

1. It is a protected method and can be called only from within the same class or a subclass.
2. You can clone only objects that are declared to implement the Cloneable interface. All arrays are considered to implement the Cloneable interface.
3. Any object whose class is Object does not implement Cloneable and throws the CloneNotSupported Exception.

Since the *clone* method is protected, this suggests packaging the public, visible *clone* method inside of the class, where it can access that protected *clone* method.

```
public abstract class SwimData implements Cloneable {

    public Object cloneMe() throws CloneNotSupportedException {
        return super.clone();
    }
```

You can also create special cloning procedures that change the data or processing methods in the cloned class, based on arguments that you pass to the *clone* method. In this case, method names such as *make* are probably more descriptive and suitable.

Using the Prototype

Now let's write a simple program that reads data from a database and then clones the resulting object. In our example program, SwimInfo, we just read these data from a file, but the original data were derived from a large database as we mentioned previously.

Then we create a class called Swimmer that holds one swimmer's name, age, sex, club, and time.

```
public class Swimmer {
    String name;    //name of swimmer
    int    age;     //age of swimmer
    String club;    //name of club
    float   time;       //result of time
    boolean female;     //sex
```