Lab 8: Overriding Object methods

Due: 11:59PM 10/17/12

Todo

Here's a class that looks familiar:

```
import java.util.Calendar;

public class Student
{
    public Student(String name, int yr, int mon, int dy)
    {
        this.name = name;
        setBirthday(yr,mon,dy);
    }

    public String getName()
    {
        return name;
    }

    public int getAge()
    {
        Calendar now = Calendar.getInstance();
        return (int)((now.getTimeInMillis() - bday.getTimeInMillis()) /
                     (365.25*24*60*60*1000));
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public void setBirthday(int yr, int mon, int dy)
    {
        bday = Calendar.getInstance();
        bday.set(yr,mon,dy);
    }
    private String name;
    private Calendar bday;
}
```

Take this class and override the following methods, inherited from `Object`:

1. `toString()` – yours should return a String containing the name and age of the student, like so: `Student[name="craig"; age=37]`
2. `equals(Object o)` – make sure both instance variables (`name` and `bday`) are considered equal
3. `clone()` – the default *shallow copy* is not good enough.

Write a driver class that demonstrates that your methods work.

Particularly, show that cloning a `Student` results in one that is `equal` to the original, but changing the birthday of the clone removes the equality.

Here's a tip for overcoming (some of) the complications of calling `super.clone()` – declare your `clone()` method like so:

```
public Object clone() throws CloneNotSupportedException
```

and your driver's `main` method like so:

```
public static void main(String[] args) throws CloneNotSupportedException
```

(We will explain what that all means in class, before long.)

Turn in

Your code in a zipped up folder named `YourName_1110_Lab8`, submitted to blackboard by the due date.