# CSC 205 Lab 6 : Our First Taste of Inheritance

*Due by Wednesday, October 4th, 11:59 PM*

## Goals

After completing this lab, you should be able to:

* Understand the role and capability of inheritance in object-oriented programming
* Use super within the constructor of a subclass, as well as other methods inherited from a parent class

## Lab Startup

Change into your Labs directory, and let's create and change into a Lab6 directory.

Now, let's copy over some files by typing : cp /pub/digh/CSC205/Lab6/\* .

You’ll want to use your class notes as a reference, as well as your text.

**The Date Class & the Object “Super” Class**

Consider the Date class which is shown on your reference sheet. Notice that the default constructor creates a GregorianCalendar object. The GregorianCalendar class is within the java.util package that allows you to access the current date and time from the operating system. Also, notice that the equals method is currently commented out.

If anyone attempts to call the equals method from a Date object, they actually will not get an error. The reason is that equals is one of the three methods common to all objects and *inherited from* the Object class (the super class of all other objects). The other two methods are clone() and toString(). The equals() method from the Object class is built to return true if the address in memory of the two objects being compared is the same. If you want equals to actually compare the contents of the data contained by an object, you have to override the default one, and write your own!

Now, let’s trace through the TestDate program which uses the Date class and predict what will be output to the screen. Write what you would expect to see printed to the screen below.

Now, let’s compile and run the TestDate program to see what actually happens. When we first compared lastSunday and holiday with the equals method, it was comparing the addresses of these objects rather than their contents. Therefore, it gave us a false. And, when we later said that

lastSunday = holiday;

we were aliasing lastSunday with holiday (i.e., giving lastSunday the same address in memory as holiday) which was what produced the true in this one.

Let’s uncomment out our equals method in our Date class, and then re-compile and re-run our TestDate program. Write your new output below.

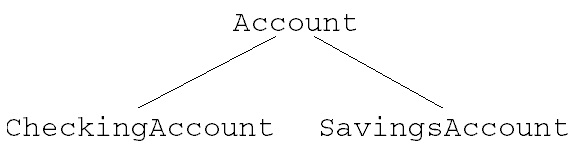
Finally, let’s do one more little experiment with our Date class. Go in and comment out the toString method. Will our TestDate program still compile? Yes! Try it and see what happens.

What’s that garbage being printed for each of your objects? Well, if you don’t explicitly define a toString() method within a class, it calls the default toString() method inherited from the Object class which merely prints some type of information about the address of the object in RAM.

Finally, go back and uncomment out your toString() method. We’ll need it again in a little bit.

**Inheriting from the Account Class**

In class, you should have seen an example of the SavingAccount class which inherits from the Account Class. Let’s now create a CheckingAccount class which also extends on the Account class. So, our inheritance diagram will look as follows.



Create the CheckingAccount class in your current directory by extending your Account class. Use your SavingAccount class example as a guide. Your CheckingAccount class will store the number of checks written against the account.

This should be an instance variable of your class.

You will need a method named getChecksWritten that returns the number of checks written so far and a method named writeCheck that writes a check for a given amount.

You’ll also need a constructor that takes a double parameter representing your balance, and sends it the parent class, Account, using super. It should also initialize your instance variable for the number of checks written to 0. When you’re done, make sure your class compiles first. Once it does, test it using the TestChecking program in your account. You should have the following output.

beg balance = $500.0

now writing a check!

total checks = 1

balance = $372.01

now closing account!

**The Employee Class (A Subclass of the Person Class)**

Take a look at the Person class for this lab. Notice that it has three instance variables : a last name, a first name, and a birth date. The names are strings, and the birth date is a Date object, where the Date is the class we used earlier.

We want to now create an Employee class that inherits from the Person class. Employee should have two instance variables : a salary (of type double) and an employment date (a Date object). Employee will need a constructor with five parameters, as well as getters and setters for the two instance variables.

It will also need a toString() method that returns a string representing a person’s name on one line, followed by their salary, birthdate, and hire date all on separate lines. In order to print a person’s name and birthdate, you’ll need to call the public methods you’ve inherited from your parent, Person. Here’s a sample of how it will print out.

name = Brown, Morris

salary = 40000.0

birth = 1980-03-08

hired = 2002-02-05

Make sure your Employee class compiles before moving onward to test it. Once it does, test it by creating a TestEmployee client program. Declare two Date objects. The first, hireDate, that gets initialized using the default Date constructor and the second, birthDate that gets initialized to March 8, 1980.

Next, declare an Employee object for the name Morris Brown with a salary of $40,000 and the hireDate and birthDate just described. Print out your object after constructing it.

**Lab #6 Reference Sheet**

**The Date Class**

public class Date {

private int year;

private int month; // 1-12

private int day; // 1-31

public Date(int year, int month, int day) {

this.year = year;

this.month = month;

this.day = day;

}

public Date() {

GregorianCalendar today = new GregorianCalendar();

year = today.get(Calendar.YEAR);

month = today.get(Calendar.MONTH) + 1;

day = today.get(Calendar.DATE);

}

public String toString() {

return year + "-" + makeTwoDigits(month) + "-" +

makeTwoDigits(day);

}

/\*

public boolean equals(Date otherDate)

{

return ( (this.year == otherDate.year) &&

(this.month == otherDate.month) &&

(this.day == otherDate.day) );

}

\*/

private static String makeTwoDigits(int n)

{

if (n <= 9)

return "0" + n;

else

return "" + n;

}

}

The Person Class

public class Person {

private String lastName;

private String firstName;

private Date birthDate;

public Person(String lastName, String firstName, Date birthDate) {

this.lastName = lastName;

this.firstName = firstName;

this.birthDate = birthDate;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public Date getBirthDate() {

return birthDate;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public void setBirthDate(Date birthDate) {

this.birthDate = birthDate;

}

}