

## Program Submission Guidelines CIS/CMSC 274

The attached packet illustrates the expected printed submissions for programming assignments (designated as Px.y in the assignment summary). The following materials are to be submitted:

1. Program scoring form using the template found on the course WebCT site (note that this form has been modified for CIS/CMSC 274 to reflect the increased importance of proper documentation using the Javadoc utility).
2. A project class diagram (most easily produced using BlueJ, but this can also be created using Violet or even MSWord).
3. One or more screen shots of program output as appropriate, inserted into an MSWord document. The illustration has been produced using BlueJ, where Alt-PrintScreen has been used to capture the window to the clipboard which was then inserted into a MSWord document. You can use the same approach for a DOS window (although I recommend you change the window display properties to be black text on a white background rather than the default white text on black background if you are using a command line interface rather than BlueJ).
4. A series of printouts of information about each class you have created or modified for your project including for each class:
  - a. The class documentation (HTML) produced by Javadoc
  - b. The corresponding java source file

You should save the MSWord document with your program output in the same project folder where the source files are located. This project folder will also contain the doc folder produced by the Javadoc utility. You can then submit all of these files by simply sending the entire project folder to me at [rmeeke@ben.edu](mailto:rmeeke@ben.edu) using e-mail (you may want to zip the folder) or submitting a copy of the project folder on suitable media (floppy disk, CD-ROM, or zip disk). Note that I expect you to print out the required material, even though I'll also be getting a copy of everything in machine-readable format. (It will only take you about 5-10 minutes to print your own project materials. If I print these myself for each student, it'll take between 90 minutes and three hours!)

PROGRAM SCORING FORM  
CIS/CMSC 274A

Program Name \_\_\_\_\_ Name \_\_\_\_\_

Course \_\_\_\_\_ Term \_\_\_\_\_

Scoring:

FIRST      SECOND  
SUBMISSION SUBMISSION

Ia — Results	(60)		
_____		_____	_____
_____		_____	_____
_____		_____	_____
_____		_____	_____

Ib — Readability	(10)		
		_____	_____

II — Program Style/Design	(10)		
		_____	_____

III — Program Documentation	(20)		
		_____	_____

TOTAL			
		_____	_____

EXTRA CREDIT			
		_____	_____

LATE PENALTY			
		_____	_____

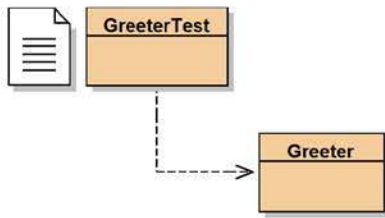
TOTAL SCORE			
	_____	+ _____	= _____ =
			2

FINAL SCORE \_\_\_\_\_

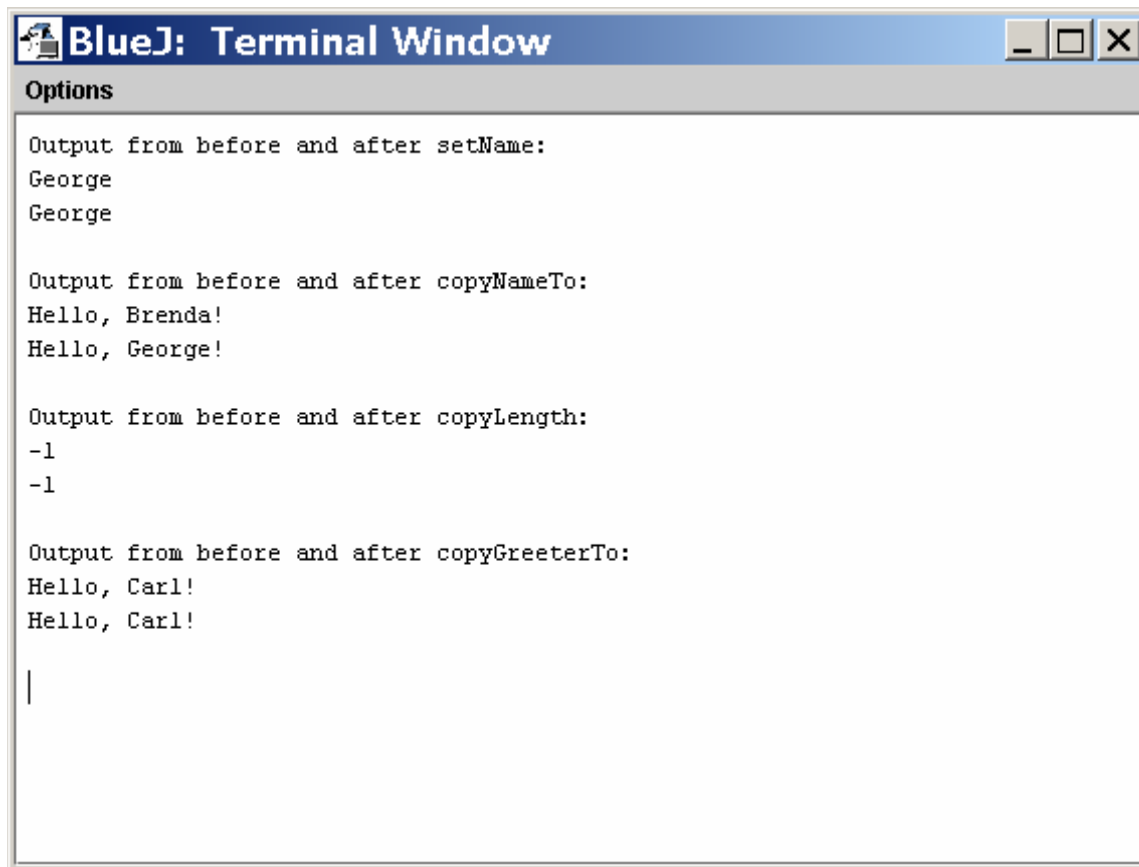
Instructor's Comments:

Redo Due: \_\_\_\_\_

# Project Ex13



Screen shot of sample program output:



The screenshot shows a window titled "BlueJ: Terminal Window" with standard window controls (minimize, maximize, close). Below the title bar is a section labeled "Options". The main area of the window contains the following text output:

```
Output from before and after setName:  
George  
George  
  
Output from before and after copyNameTo:  
Hello, Brenda!  
Hello, George!  
  
Output from before and after copyLength:  
-1  
-1  
  
Output from before and after copyGreeterTo:  
Hello, Carl!  
Hello, Carl!  
  
|
```

## Class GreeterTest

[java.lang.Object](#)

|  
+--**GreeterTest**

```
public class GreeterTest
    extends Object
```

A class for testing the effects on parameters of the methods in section 1.6

### Constructor Summary

[GreeterTest](#)()

### Method Summary

static void	<a href="#">main</a> ( <a href="#">String</a> [] args) Creates a greeter object then calls several methods and prints the values of the parameter variables before and after each call.
-------------	--

### Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

### Constructor Detail

#### GreeterTest

```
public GreeterTest()
```

### Method Detail

#### main

```
public static void main(String[] args)
    Creates a greeter object then calls several methods and
    prints the values of the parameter variables before and
    after each call.
Parameters:
    args - unused
```



```
/**
 * A class for testing the effects on parameters of the methods in section 1
 * .6
 */
public class GreeterTest
{
    /**
     * Creates a greeter object then calls several methods and prints the
     * values of the parameter variables before and after each call.
     * @param args unused
     */
    public static void main(String[] args)
    {
        Greeter g1 = new Greeter("Adam");

        System.out.println("Output from before and after setName:");
        String aName = "George";
        System.out.println(aName);
        g1.setName(aName);
        System.out.println(aName);
        System.out.println();

        System.out.println("Output from before and after copyNameTo:");
        Greeter g2 = new Greeter("Brenda");
        System.out.println( g2.sayHello() );
        g1.copyNameTo(g2);
        System.out.println( g2.sayHello() );
        System.out.println();

        System.out.println("Output from before and after copyLength:");
        int x = -1;
        System.out.println( x );
        g1.copyLengthTo(x);
        System.out.println( x );
        System.out.println();

        System.out.println("Output from before and after copyGreeterTo:");
        Greeter g3 = new Greeter("Carl");
        System.out.println( g3.sayHello() );
        g1.copyGreeterTo(g3);
        System.out.println( g3.sayHello() );
        System.out.println();
    }
}
```

## Class Greeter

[java.lang.Object](#)

|  
+--**Greeter**

```
public class Greeter
    extends Object
```

A class for producing simple greetings. (Revised to include methods from section 1.6)

### Constructor Summary

[Greeter](#)([String](#) aName)

Constructs a Greeter object that can greet a person or entity.

### Method Summary

void [copyGreeterTo](#)([Greeter](#) other)

Tries to set another Greeter object to a copy of this object

void [copyLengthTo](#)(int n)

Tries to copy the length of this greeter's name into an integer variable

void [copyNameTo](#)([Greeter](#) other)

Sets another greeter's name to this Greeter's name.

[String](#) [sayHello](#)()

Greet with a "Hello" message.

void [setName](#)([String](#) name)

Sets this greeter's name to the given name.

### Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

### Constructor Detail

#### Greeter

```
public Greeter(String aName)
```

Constructs a Greeter object that can greet a person or entity.

**Parameters:**



aName - the name of the person or entity who should be addressed in the greetings.

## Method Detail

### copyGreeterTo

```
public void copyGreeterTo(Greeter other)
    Tries to set another Greeter object to a copy of this object
Parameters:
    other - the Greeter object to initialize
```

---

### copyLengthTo

```
public void copyLengthTo(int n)
    Tries to copy the length of this greeter's name into an
    integer variable
Parameters:
    n - the the variable into which the method tries to copy the
    length
```

---

### copyNameTo

```
public void copyNameTo(Greeter other)
    Sets another greeter's name to this Greeter's name.
Parameters:
    other - a reference to the other Greeter
```

---

### sayHello

```
public String sayHello()
    Greet with a "Hello" message.
Returns:
    a message containing "Hello" and the name of the greeted
    person or entity.
```

---

### setName

```
public void setName(String name)
    Sets this greeter's name to the given name.
Parameters:
    name - the new name for the object
```

---

Package [Class](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

---

```
/**
 * A class for producing simple greetings. (Revised to include methods
 * from section 1.6)
 */
public class Greeter
{
    /**
     * Constructs a Greeter object that can greet a person or
     * entity.
     * @param aName the name of the person or entity who should
     * be addressed in the greetings.
     */
    public Greeter(String aName)
    {
        name = aName;
    }

    /**
     * Greet with a "Hello" message.
     * @return a message containing "Hello" and the name of
     * the greeted person or entity.
     */
    public String sayHello()
    {
        return "Hello, " + name + "!";
    }

    /**
     * Sets this greeter's name to the given name.
     * @param name the new name for the object
     */
    public void setName(String name)
    {
        this.name = name;
    }

    /**
     * Sets another greeter's name to this Greeter's name.
     * @param other a reference to the other Greeter
     */
    public void copyNameTo(Greeter other)
    {
        other.name = this.name;
    }

    /**
     * Tries to copy the length of this greeter's name into an integer variab
le
     * @param n the the variable into which the method tries to copy the leng
```

```
th
    */
    public void copyLengthTo(int n)
    {
        n = name.length();
    }

    /**
     * Tries to set another Greeter object to a copy of this object
     * @param other the Greeter object to initialize
     */
    public void copyGreeterTo(Greeter other)
    {
        other = new Greeter(name);
    }

    private String name;
}
```