Unit Eight Lecture One

Topic 1: Code Reuse

A characteristic of all good computer programmers is to never rewrite code you have already developed. This can mean within a single project or across a lifetime of projects.

To accomplish this goal, programmers can resort to two completely different approaches.

The first approach, which most novice programmers pick up on quickly, is to copy and paste code from one application to another. The only requirement for this approach is that you store your program files in some orderly fashion so that you can easily put your hands on a code segment that you need to duplicate.

The second approach, which is much more complex than the first, requires the programmer to learn how to extend the use of existing classes. The concepts I'm referring to include interfaces & polymorphism as well as inheritance.
We are going to cover interfaces & polymorphism this unit and inheritance in the next.

To fully appreciate how interfaces & polymorphism can assist the programmer in reducing duplicated code we should first study a project that would be appropriate for an interface but doesn't implement it.

In this project, we do two similar statistical analyses on the values obtained by the instance variable *ppg* of the BBPlayer class and the values obtained by the instance variable *balance* of the BankAccount class. As you recall, the values of *ppg* were obtained from the text file U7Test.txt and the values of *balance* were obtained from the text file myCreditUnion.txt. For each of these two instance fields, we need to determine minimum value, maximum value, and the average value sent to the class.

In this project, we can assume the BBPlayer class and the BankAccount class were not changed.

Next, we develop a class called DataSet1 designed to handle the BBPlayer class. It should have 5 private static instance fields: max, min, ave, sum, & count. In the default constructor, assign all 5 fields a value of 0.0. The parametric constructor should accept a variable of type BBPlayer, access the value of the instance field ppg, increment count by one, accumulate ppg into sum, test ppg against max

and min, and determine current average. This class must also have accessor methods getMax(), getMin(), and getAve().

Next, we develop a class called DataSet2 designed to handle the BankAccount class. This class is very much like DataSet1. The parametric constructor must accept a variable of type BankAccount and access the value of the instance field balance.

[NOTE: The recommended procedure to find max and min is to assign to both of them the first number processed. Succeeding numbers are then compared to max and min and the values of max and min are changed if necessary.]

Finally, we develop the driver class that extends JFrame. It must test DataSet1 by sending it objects of type BBPlayer containing the data stored in U7Test.txt. It must also test DataSet2 by sending it objects of type BankAccount containing the data stored in myCreditUnion.txt.

The code and output of this project can be found on the following pages.

```
// This is the code for a DataSet class specifically designed to handle
// the BBPlayer class.

public class DataSet1
{
	private static double max;
	private static double min;
	private static double ave;
	private static double sum;
	private static int count;

	public DataSet1()
	{
		max = 0;
		min = 0;
		ave = 0;
		sum = 0;
		count = 0;
	}
	public DataSet1(BBPlayer x)
	{
		count++;
		sum += x.getPPG();
		if (count == 1)
		{
			max = x.getPPG();
			min = x.getPPG();
		}
		else
		{
			if (x.getPPG() > max)
				max = x.getPPG();
			if (x.getPPG() < min)
				min = x.getPPG();
		}
		ave = sum/count;
	}
	public double getMax()
	{
		return max;
	}
	public double getMin()
	{
		return min;
	}
	public double getAve()
	{
		return ave;
	}
}
```

```
// This is the code for a DataSet2 class specifically designed to
// handle the BankAccount class.

public class DataSet2
{
	private static double max;
	private static double min;
	private static double ave;
	private static double sum;
	private static int count;

	public DataSet2()
	{
		max = 0;
		min = 0;
		ave = 0;
		sum = 0;
		count = 0;
	}
	public DataSet2(BankAccount x)
	{
		count++;
		sum += x.getNumBalance();
		if (count == 1)
		{
			max = x.getNumBalance();
			min = x.getNumBalance();
		}
		else
		{
			if (x.getNumBalance() > max)
				max = x.getNumBalance();
			if (x.getNumBalance() < min)
				min = x.getNumBalance();
		}
		ave = sum/count;
	}
	public double getMax()
	{
		return max;
	}
	public double getMin()
	{
		return min;
	}
	public double getAve()
	{
		return ave;
	}
```

```
// This driver class will try to prepare students for the concept of an
// interface

import javax.swing.JFrame;
import javax.swing.JTextArea;
import java.awt.Container;
import java.awt.Font;
import java.io.*;
import java.util.StringTokenizer;
import java.util.Scanner;

public class u8a1 extends JFrame
{
	private JTextArea output = new JTextArea();

	public static void main(String args[])
	{
		u8a1 x = new u8a1();
		x.setSize(300,500);
		x.setVisible(true);
		x.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
	public u8a1()
	{
		TestBBPlayer();
		TestBankAccount();
	}
	public void TestBBPlayer()
	{
		String input;
		DataSet1 z = new DataSet1();
		try
		{
			Scanner in = new Scanner(new File("u7test.txt"));
			for (int i=0; i<5; i++)
			{
				input = in.nextLine();
				BBPlayer aplayer = new BBPlayer(input);
				z = new DataSet1(aplayer);
			}
			in.close();
		}
		catch (IOException e)
		{
			throw new RuntimeException(e.toString());
		}
		Container container = getContentPane();
		output.setFont(new Font("Monospaced", Font.BOLD, 14));
		output.setText("BBPlayer Stats\n");
		output.append("--------------\n\n");
		double bbmin, bbmax, bbave;
		bbmin = z.getMin();
		bbmax = z.getMax();
		bbave = z.getAve();

		output.append("Minimum Points Per Game = " + bbmin + "\n\n");
		output.append("Maximum Points Per Game = " + bbmax + "\n\n");
		output.append("Average Points Per Game = " + bbave + "\n\n");
		container.add(output);
	}
```

```
	public void TestBankAccount()
	{
		String num, sbal;
		double bal;
		DataSet2 z = new DataSet2();
		try
		{
			Scanner in = new Scanner(new File("myCreditUnion.txt"));
			while (in.hasNext())
			{
				String line = in.nextLine();
				StringTokenizer tokenizer = StringTokenizer(line);
				num = tokenizer.nextToken();
				sbal = tokenizer.nextToken();
				bal = Double.parseDouble(sbal);
				BankAccount anAccount = new BankAccount(num, bal);
				z = new DataSet2(anAccount);
			}
			in.close();
		}
		catch (IOException e)
		{
			throw new RuntimeException(e.toString());
		}
		output.append("\nBankAccount Stats\n");
		output.append("-----------------\n\n");
		double bamin, bamax, baave;
		bamin = z.getMin();
		bamax = z.getMax();
		baave = z.getAve();
		output.append("Minimum Bank Account = " + bamin + "\n\n");
		output.append("Maximum Bank Account = " + bamax + "\n\n");
		String x = String.format("%.2f", baave);
		output.append("Average Bank Account = " + x + "\n\n");

	}
}
```