Lab 6

(Due date: Wednesday, 2/18/2009 in the Lab hours)

Your name:	Score:

1. For each of the following Java projects:
   a. **Read** the following complete Java code;
   b. **Write down** in the blank space the Java statements which are still new/difficult for you to use them in your own code;
   c. **Figure out** what this code is trying to do (by **providing several sets of inputs and corresponding outputs**).
   d. **Test** the Java code to see if you have correctly understood the code by running the Java code (the code is available from R: Drive).

1a). Fig4.6-7:

```
//Fig4.7
public class GradeBookTest
{
   public static void main( String args[] )
   {
      GradeBook myGradeBook = new GradeBook(
         "CS101 Introduction to Java Programming" );

      myGradeBook.displayMessage();
     myGradeBook.determineClassAverage();
   }
}

//Fig4.6
import java.util.Scanner; // program uses class Scanner

public class GradeBook
{
   private String courseName;
   public GradeBook( String name )
   {
      courseName = name;
   }
   public void setCourseName( String name )
   {
      courseName = name; // store the course name
   }
   public String getCourseName()
   {
      return courseName;
   }
   public void displayMessage()
   {
      System.out.printf( "Welcome to the grade book for\n%s!\n\n",
         getCourseName() );
   }
   public void determineClassAverage()
   {
```

```
      Scanner input = new Scanner( System.in );

      int total; // sum of grades entered by user
      int gradeCounter; // number of the grade to be entered next
      int grade; // grade value entered by user
      int average; // average of grades

      total = 0; // initialize total
      gradeCounter = 1; // initialize loop counter

      while ( gradeCounter <= 10 ) // loop 10 times
      {
         System.out.print( "Enter grade: " ); // prompt
         grade = input.nextInt(); // read grade from user
         total = total + grade; // add grade to total
         gradeCounter = gradeCounter + 1; // increment counter by 1
      }

      average = total / 10; // integer division yields integer result

      System.out.printf( "\nTotal of all 10 grades is %d\n", total );
      System.out.printf( "Class average is %d\n", average );
   }
}
```

1b). Fig4.9-10.

```
// Fig. 4.10: GradeBookTest.java
// Create GradeBook object and invoke its determineClassAverage method.

public class GradeBookTest
{
   public static void main( String args[] )
   {
      // create GradeBook object myGradeBook and
      // pass course name to constructor
      GradeBook myGradeBook = new GradeBook(
         "CS101 Introduction to Java Programming" );

      myGradeBook.displayMessage(); // display welcome message
      myGradeBook.determineClassAverage(); // find average of grades
   } // end main

} // end class GradeBookTest

// Fig. 4.9: GradeBook.java
// GradeBook class that solves class-average program using
// sentinel-controlled repetition.
import java.util.Scanner; // program uses class Scanner

public class GradeBook
{
   private String courseName; // name of course this GradeBook represents

   // constructor initializes courseName
   public GradeBook( String name )
   {
```

```
   courseName = name; // initializes courseName
} // end constructor

// method to set the course name
public void setCourseName( String name )
{
   courseName = name; // store the course name
} // end method setCourseName

// method to retrieve the course name
public String getCourseName()
{
   return courseName;
} // end method getCourseName

// display a welcome message to the GradeBook user
public void displayMessage()
{
   // getCourseName gets the name of the course
   System.out.printf( "Welcome to the grade book for\n%s!\n\n",
      getCourseName() );
} // end method displayMessage

// determine the average of an arbitrary number of grades
public void determineClassAverage()
{
   // create Scanner to obtain input from command window
   Scanner input = new Scanner( System.in );

   int total; // sum of grades
   int gradeCounter; // number of grades entered
   int grade; // grade value
   double average; // number with decimal point for average

   // initialization phase
   total = 0; // initialize total
   gradeCounter = 0; // initialize loop counter

   // processing phase
   // prompt for input and read grade from user
   System.out.print( "Enter grade or -1 to quit: " );
   grade = input.nextInt();

   // loop until sentinel value read from user
   while ( grade != -1 )
   {
      total = total + grade; // add grade to total
      gradeCounter = gradeCounter + 1; // increment counter

      // prompt for input and read next grade from user
      System.out.print( "Enter grade or -1 to quit: " );
      grade = input.nextInt();
   } // end while

   // termination phase
   // if user entered at least one grade...
   if ( gradeCounter != 0 )
   {
      // calculate average of all grades entered
      average = (double) total / gradeCounter;
```

```
         // display total and average (with two digits of precision)
         System.out.printf( "\nTotal of the %d grades entered is %d\n",
            gradeCounter, total );
         System.out.printf( "Class average is %.2f\n", average );
       } // end if
       else // no grades were entered, so output appropriate message
         System.out.println( "No grades were entered" );
    } // end method determineClassAverage

} // end class GradeBook
```

1c). Fig4.12-13

```
// Fig. 4.13: AnalysisTest.java
// Test program for class Analysis.

public class AnalysisTest
{
   public static void main( String args[] )
   {
      Analysis application = new Analysis(); // create Analysis object
      application.processExamResults(); // call method to process results
   } // end main

} // end class AnalysisTest


// Fig. 4.12: Analysis.java
// Analysis of examination results.
import java.util.Scanner; // class uses class Scanner

public class Analysis
{
   public void processExamResults()
   {
      // create Scanner to obtain input from command window
      Scanner input = new Scanner( System.in );

      // initializing variables in declarations
      int passes = 0; // number of passes
      int failures = 0; // number of failures
      int studentCounter = 1; // student counter
      int result; // one exam result (obtains value from user)

      // process 10 students using counter-controlled loop
      while ( studentCounter <= 10 )
      {
         // prompt user for input and obtain value from user
         System.out.print( "Enter result (1 = pass, 2 = fail): " );
         result = input.nextInt();

         // if...else nested in while
         if ( result == 1 )          // if result 1,
            passes = passes + 1;     // increment passes;
         else                        // else result is not 1, so
            failures = failures + 1; // increment failures

         // increment studentCounter so loop eventually terminates
         studentCounter = studentCounter + 1;
      } // end while
```

```
        // termination phase; prepare and display results
        System.out.printf( "Passed: %d\nFailed: %d\n", passes, failures );

        // determine whether more than 8 students passed
        if ( passes > 8 )
           System.out.println( "Raise Tuition" );
     } // end method processExamResults

  } // end class Analysis
```

1d). Fig4.16

```
// Fig. 4.16: Increment.java
// Prefix increment and postfix increment operators.

public class Increment
{
   public static void main( String args[] )
   {
      int c;

      // demonstrate postfix increment operator
      c = 5; // assign 5 to c
      System.out.println( c );   // prints 5
      System.out.println( c++ ); // prints 5 then postincrements
      System.out.println( c );   // prints 6

      System.out.println(); // skip a line

      // demonstrate prefix increment operator
      c = 5; // assign 5 to c
      System.out.println( c );   // prints 5
      System.out.println( ++c ); // preincrements then prints 6
      System.out.println( c );   // prints 6

   } // end main

} // end class Increment
```

2. **Complete Exercise 3.13 on page 125—print out and submit your code.**

3. **Complete GUI and Graphics Case Study Exercise 3.1 on page 111—print out and submit your code.**

4. **Complete Exercise 3.14 on page 126—print out and submit your code.**