Name: ____________________________________________

NetID: __________

Statement of integrity: I did not, and will not, violate the rules of academic integrity on this exam.

(Signature)

Circle your lecture time: 9:05 or 11:15

Q1: _____ 20pts ____
Q2: _____ 20pts ____
Q3: _____ 40pts ____
Q4: _____ 20pts ____

Total: _____ 100pts ____

Instructions:
- This is a 90-minute, closed-book exam; no calculators are allowed.
- There are 4 questions worth a total of 100 points ⇒
  A rough time budget: Try not to spend more than 18 minutes on a 20-point question.
- Raise your hand if you have any questions.
- Use the backs of pages or ask for additional sheets of paper as necessary.
- Clarity, conciseness, and style count for credit.
- If you supply multiple answers, we will grade only one.
- Use only Java code. No credit for code written in other programming languages.
- Do not use switch, break, or System.exit statements.
- Do not use arrays.
Question 1: (20 points)

Part (a): (6 points)
Define “method signature.” Be concise. __________________________________________________________  
____________________________________________________________________________________

Part (b): (11 points)
Consider class Q1b below. The specification and header for method timeInSec is shown but the method body is hidden. Assume that method timeInSec has been correctly implemented—it can be called from method main. The numbered statements are attempts to call method timeInSec. Write on the blank following each call the word “good” if the call is correctly written or the word “bad” if the call is incorrect.

```java
public class Q1b {
    /** = number of seconds from 00:00:00 to h:m:s. Return whole seconds only. */
    public static int timeInSec(int h, int m, double s) {
        // Code not shown. Assume method is implemented correctly.
        // ...
    }

    public static void main(String[] args){
        int h=20, m=58; //hour, minute
        double s=12.6;   //second
        int sec;

        sec= timeInSec(20, 58, 12.6);    //1. ________________________
        sec= timeInSec(h, m, s);    //2. ________________________
        sec= timeInSec(h, m, (int) s);    //3. ________________________
        double d= timeInSec(h, m, (int) s);    //4. ________________________
        sec= timeInSec(h, (int) Math.round(m + s/60));  //5. ________________________
        sec= timeInSec(new String(h+":"+m+":"+s));  //6. ________________________
        System.out.println(timeInSec(20, 58, s));  //7. ________________________
        sec= Q1b.timeInSec(20, 58, s);    //8. ________________________
        sec= class.timeInSec(20, 58, s);    //9. ________________________
        sec= static.timeInSec(20, 58, s);    //10. ________________________
        sec= this.timeInSec(20, 58, s);    //11. ________________________
    }
}
```

Part (c): (3 points)
Our textbook (and Program Live) discusses a technique used in programming that was used also by Edgar Allan Poe in writing his poem The Raven. What is this technique?
Question 2: (20 points)

Consider the sequence

\[ 1, 2, -3, 4, 5, -6, 7, 8, -9, \ldots \]

Given \( n > 0 \), write a program fragment to display the first \( n \) terms of the sequence in reverse order. Also display the sum of the sequence. For example, if \( n \) is 4, the sequence displayed should be:

\[
\begin{align*}
4 & \\
-3 & \\
2 & \\
1 &
\end{align*}
\]

and the sum of the sequence is \( 4 + (-3) + 2 + 1 = 4 \). Do not use arrays.

```java
public class Q2 {
    public static void main(String[] args) {
        System.out.println("Enter a positive integer: ");
        int n = JLiveRead.readInt();  // Number of terms in the sequence to print
        // Assume n is positive

        int sum;                      // Sum of the first n terms in the sequence

        // Display the first n terms of the sequence in reverse order and calculate the sum.

        System.out.println("The sum of the first " + n + " terms is "+ sum);
    } // method main
} // class Q2
```
Question 3: (40 points)

Complete classes Rectangle and Q3 below. Class Rectangle represents a rectangle and has the following variables and methods:

- Instance variables width and height: the width and height (type double) of a Rectangle
- A constructor that has two parameters: double width, double h
- Instance method area() returns the area (type double) of the current Rectangle
- Instance method isSquare() returns true if the current Rectangle is a square, false otherwise
- Instance method cutHalf() cuts off vertically half of the current Rectangle
- Instance method toString() gives a String description of the dimensions (width and height) and area of the current Rectangle
- Class method average(Rectangle r1, Rectangle r2) returns a new Rectangle with dimensions that are the average values between Rectangles r1 and r2

Do not define any other instance or class variables/methods

Class Q3 is a client class of Rectangle. Class Q3 has a single method main where you will create one Rectangle. Then you will repeatedly cut the Rectangle in half until it becomes a square or its area is less than a specified value. Print the information of the final Rectangle.

Read through both incomplete classes before you start writing. Follow the specifications above and in the comments. You must use the variable and parameter names and types as specified above. Use encapsulation (use the modifiers private and public appropriately). To indicate that a blank (or box) should be left empty, draw a diagonal line across the blank or box.

```java
/** A rectangle */
class Rectangle {

    ____________________________ double width;    //width of the Rectangle
    ____________________________ double height;   //height of the Rectangle

    /** Constructor: assign values to the fields */
    _______________________________________________________________________
    (double width , double h ){

    }

    /** = Get area of this Rectangle */
    _______________________________________________________________________
    area() {

    }

    //Class Rectangle continues on next page
```
//Class Rectangle, continued (Question 3, continued)

/** = This rectangle is a square */

isSquare() {
}

/** Cut off vertically half of this Rectangle. I.e., reduce the width by half */
cutHalf() {
}
/** = String description of the dimensions and area of this Rectangle */
toString() {
}

/** A Class method. = Get a new Rectangle whose width is the average width * between r1 and r2 and whose height is the average height between r1, r2 */
average(Rectangle r1, Rectangle r2) {
}

} //class Rectangle

//Question 3 continues on next page
/* Class Q3, client of class Rectangle (Question 3, continued) */
public class Q3 {
    public static void main(String[] args) {
        //Create a Rectangle object, use reference variable rec:
        double w  = JLiveRead.readDouble();  // width of rec
        double h  = JLiveRead.readDouble();  // height of rec

        //Repeatedly cut rec in half until it becomes a square OR until its area
        //is less than MINarea. Display Rectangle rec’s data at the end.
        final double MINarea= 10;
    } //method main
} //class Q3
Question 4: (20 points)

Typically, a student has a bursar account representing the amount that she or he owes the university. **Design** a class `Account` whose instances represent students’ bursar accounts. An account is associated with a student name and a student ID and has a balance. It should be possible to retrieve these values. Furthermore, it should be possible to charge to (increase balance of) and make payment to (decrease balance of) the account. It should be possible to determine if the account has an owing balance so that a statement can be printed (showing the student name, ID, and balance).

Design the class by writing variable declarations and method specifications and headers. Use *meaningful* variable and method names. Specifications (comments) must be *concise*. Do **not** write the method bodies!