Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided.

1. (10 points) True/False (**T** or **F**)

_____ If `import` javax.swing.JOptionPane is not present before the class declaration, all occurences of JOptionPane must be replaced with javax.swing.JOptionPane.

_____ Math.cos(90) calculates the cosine of 90 degrees.

_____ For double x = Math.random() * 3 - 1, $-1.0 \leq x < 2.0$.

_____ JOptionPane.showMessageDialog(null, "Hi"); is an example of a call to a `static` method.

_____ A `private` attribute is accessible to all methods defined within the class.

_____ An accessor method should be declared as `private`.

_____ The source code for all classes in the same package must be located in the same subdirectory/folder.

_____ Camel case refers to a technique involving the `switch` statement.

_____ The MSOE coding standard specifies the order in which information is included in Java source files.

_____ According to the MSOE coding standard, .java filenames must begin with a capital letter.

2. (10 points) In the code below, place an oval around every primitive and a rectange around every object reference.

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter a complex number (e.g., 3.0 + 4.3i)");
    String line = in.nextLine();
    Scanner parser = new Scanner(line.substring(0, line.length()-1));
    double real = parser.nextDouble();
    parser.next();
    double imaginary = parser.nextDouble();
    double mag = Math.sqrt(Math.pow(real, 2) + imaginary * imaginary);
    double angle = Math.toDegrees(Math.atan(imaginary / real));
    System.out.println("You entered " + mag + " | " + angle);
}
```

3. (10 points) Clearly and succinctly describe how an object differs from a primitive.

4. (10 points) Give an example of a method implementation that makes use of the keyword: `this` to disambiguate between a local variable and an instance variable.

```
public class Exam2 {
    private boolean isEasy;

}
```

5. (10 points) Clearly and succinctly explain the differences between a **local variable**, **instance variable**, and **class variable**.

6. (15 points)

```
public static void main(String[] args) {
    Fraction f1 = new Fraction(1, 3);
    Fraction f2 = new Fraction(2, 5);
    Fraction f3 = f1.divideBy(f2);
    Fraction f4 = f1.times(f3);
    System.out.println(f1.toString() + "␣/␣" + f2.toString() + "␣=␣" + f3.toString());
    System.out.println(f1.toString() + "␣*␣" + f3.toString() + "␣=␣" + f4.toString());
```

The above code produces the following output:

```
1/3 / 2/5 = 5/6
1/3 * 5/6 = 5/18
```

Draw the UML class diagram for the Fraction class used in the code above.

7. (15 points)
The code below is identical to the previous problem and is reproduced here for your convenience.

```
public static void main(String[] args) {
    Fraction f1 = new Fraction(1, 3);
    Fraction f2 = new Fraction(2, 5);
    Fraction f3 = f1.divideBy(f2);
    Fraction f4 = f1.times(f3);
    System.out.println(f1.toString() + " / " + f2.toString() + " = " + f3.toString());
    System.out.println(f1.toString() + " * " + f3.toString() + " = " + f4.toString());
```

The above code produces the following output:

```
1/3 / 2/5 = 5/6
1/3 * 5/6 = 5/18
```

Draw the memory diagram (like the diagrams drawn in lecture) that illustrates the state of memory after the first println() statement has completed.

8. (20 points)
The code below is identical to the previous problem and is reproduced here for your convenience.

```
public static void main(String[] args) {
    Fraction f1 = new Fraction(1, 3);
    Fraction f2 = new Fraction(2, 5);
    Fraction f3 = f1.divideBy(f2);
    Fraction f4 = f1.times(f3);
    System.out.println(f1.toString() + " / " + f2.toString() + " = " + f3.toString());
    System.out.println(f1.toString() + " * " + f3.toString() + " = " + f4.toString());
```

The above code produces the following output:

```
1/3 / 2/5 = 5/6
1/3 * 5/6 = 5/18
```

Implement the complete Fraction class used in the code above. Your implementation should cause the main method to compile and execute without errors and produce the sample output shown above. Your implementation does not need to reduce fractions to their simplest form or handling negative values in the denominator.

Additional space — indentify which problem your work is associated with.