**CS100J Spring 2004: Project 2 Grading Guide**

**Scores**

- c and s stand for correctness and style; see table below.
- ** counts as two items (errors)
- +x is x bonus points for exemplary work (contributes to semester "Bonus Bucket")

| Score | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| # correctness errors | Nothing turned in | >=10 | 7-9 | 5-6 | 3-4 | 0-2 |
| # style errors | Nothing turned in | >=7 | 4-6 | 2-3 | 1 | 0 |

**General**

(s0a) proper code indentation
(s0b) using named constants, appropriate comments, and meaningful variable names
(s0c) line length limited to 70 characters
(s0d) appropriate code header comments
(c0a) q1 and q5 is submitted as a plain text file "q1q5.txt"

**1. Warm Up Questions**

(c1a) a and b are non-static
(c1b) modifier added is public
(c1c) filename is ComplexNumber.java
(c1d) ComplexNumber is not an application

**2. ComplexNumber class**

(c2a) a and b are doubles
(c2b) a and b are public, non-static
(c2c) Constructor sets arguments correctly
(c2d) add method adds correctly
(c2e) subtract method subtracts correctly
(c2f) multiply method multiplies correctly
(c2g) divide method divides correctly
(c2h) magnitude method returns correct magnitude
(c2i) conjugate method returns correct conjugate
(c2j) isComplex method correctly identifies complex number
(c2k) toString handles real-only numbers
(c2l) toString handles positive imaginary numbers
(c2m) toString handles negative imaginary numbers (no plus sign)
(c2n) toString handles complex-only numbers
(c2o) No compilation errors
(c2p) Correct method declarations
(c2q) Unsuccessful attempt to use DecimalFormat should not interfere with toString's correctness
(s2a) Co mments for each field
(s2b) Specification comments for each method
(s3c) Helpful comments in toString

**3. ComplexCircle**

(c3a) ** Uses a loop (otherwise, this is unacceptable inefficient; what if we had asked for 5 degree increments instead of 45 degree increments?)
(c3b) Converts to radians at some stage
(c3c) Increments angle appropriately
(c3d) Displays correct corresponding complex number
(c3e) no compilation error
(c3f) Correct method declaration
(s3a) Specification comment for the method
(s3b) Helpful comments in method
(s3c) Displays each complex number in some delimited manner (i.e. Easy to read)

(s3d) Reuses existing magnitude code

**4. Client Code**
(c4a) Correctly prompts for the real part of the complex number
(c4b) Correctly prompts for the imaginary part of the complex number
(c4c) Correctly constructs complex number based on user input
(c4c) Uses an if-else if-else block
(c4d) Uses correct conditions
(c4e) Calls each Op method correctly
(c4f) no compilation error
(c4g) Loop terminates only when user chooses to
(s4a) no changes in framework code
(s4b) prompts to user are meaningful

**5. Testing your code**
(c5a) Answers are different because of double precision/round off error
(c5b) The loop terminates when choice == 8 and choice is user determined

**6. Bonus**
(+1) ComplexNumber toString method prints formatted string with one decimal place. *Note if not using DecimalFormat class, student must take care to implement half-even rounding.
(+1) CompexNumber divide method handles divide by zero cases