UNIVERSITY OF WISCONSIN-LA CROSSE
Department of Computer Science

CS 120	Software Design I	Fall 2017
Midterm Exam 01		09 October 2017

- **Do not turn the page until instructed to do so.**
- **This booklet contains 10 pages including the cover page.**
- **This is a closed-book exam. All you need is the exam and a writing utensil. (You may use a calculator if you wish.)**
- **You have exactly 55 minutes.**
- **The maximum possible score is 50.**

PROBLEM	SCORE
1	
2	
3	
4	
5	
6	
TOTAL	

NAME: ______________________________

1. **(8 pts.)** **TRUE/FALSE.**

For each of the following, indicate whether the statement is true or false.
You do not need to explain your answers.

a. If the left-hand target of an assignment statement is an `int` variable, then the expression on the right-hand side will always be evaluated using integer arithmetic.

b. The following two lines of code produce the same result:

```
String str = 3 + "4" + 5;
String str = "3" + 4 + 5;
```

c. The following two lines of code produce the same result:

```
String str = 3 + 4 + "5";
String str = "3" + 4 + 5;
```

d. When doing arithmetic in Java, cast operations have ***higher*** precedence than the additive operators (`+`, `-`), but ***lower*** precedence than the multiplicative ones (`*`, `/`).

e. The following line of code will not compile, due to an error:

```
int x = 1 / 2;
```

f. The following line of code will not compile, due to an error:

```
String x = 1 / 2;
```

g. The following will always print some result, ***no matter what*** value the integer `x` has (assuming that it has some value):

```
if( x > 0 ) {
    System.out.println( "First" );
}
else if ( x == 0 ) {
    System.out.println( "Second" );
}
```

h. The following will always print some result, ***no matter what*** value the integer `x` has (assuming that it has some value):

```
if( x > 0 ) {
    System.out.println( "First" );
}
else {
    System.out.println( "Second" );
}
```

2. **(8 pts.)** **SHORT ANSWER.**

Circle the appropriate answer and fill in the blanks where required.

a. (*4 pts.*) An assignment like the following:

```
int x = (int)( 13 / 2.0 );
```

(WILL / WON'T) compile properly, because ______________________________

____.

An assignment like the following:

```
int x = (int) 13 / 2.0;
```

(WILL / WON'T) compile properly, because ______________________________

____.

b. (*4 pts.*) Consider the following method description, from the `String` class:

```
public String substring( int start, int end )
```

This is a **(VOID / NON-VOID)** method, which has ______________________________

as output. Inside the parentheses, the information `( int start, int end )` tells us that

the method ____

____.

3. **(10 pts.)** **CODE EVALUATION (I).**

For each of the following, give the value of the variable `x` after each line executes. If the line produces an error, then write `ERROR`. If the variable can have different values (as when using a random number generator), then indicate those values by writing, e.g., `1 <= x <= 5`.

a. `int x = 10 - 5 * 3 / 2;`

b. `int x = (int)( 10 - 5 * 3 / 2.0  );`

c. `int x = 10 - 5 * (int)( 3 / 2.0 );`

d. `int x = 10 - 5 * 3 / (int) 2.0;`

e. `double x = 30 + 7 / 2;`

f. `double x = 30 + 7 / 2.0;`

g.
```
String s = "Banana!";
String x = s.substring( 1, 4 );
```

h. `int x = (int)( Math.random() * 30 + 1 );`

i. `int x = (int)( Math.random() * 10 + 10 );`

j. `boolean x = ( 5 == 5.0 );`

4. **(4 pts.)** **CODE EVALUATION (II).**

Consider the following code:

```
String s1 = "This";
String s2 = "Is";
String s3 = "A";
String s4 = "Test";

s3 = s1;
s1 = s4;
s2 = s4;
s4 = s3;
```

a. The first four lines of code above create 4 distinct `String` objects in memory. When the code is complete, ***two*** of those objects have been orphaned. Write down the lines of code that causes this to happen, and write down the `String` literals that are orphaned by each.

b. Circle any method call below that would produce the result `true`, if executed ***after all*** of the code given in the question is complete. Remember that the `equals()` method returns `true` if and only if both the `String` that is calling the method and the input `String` are the same, character for character:

(i) `s1.equals( s3 )`

(ii) `s3.equals( s4 )`

(iii) `s2.equals( s4 )`

(iv) `s4.equals( s1 )`

5. **(10 pts.)** **CODE COMPLETION (I).**

On the next page, fill in the class given so that it contains a `main()` method that:

a. Uses `System.out` to ask the user for an integer value from `1` to `10` (inclusive), and reads it in from `System.in`, using a `Scanner`.

b. Generates its own random number from `1` to `10` (inclusive), and tells the user if their guess was correct, too high, or too low, while also showing the random number that was generated whenever they guess incorrectly. (See below for expected format.)

c. Tells the user that they have made an improper guess if they either (i) enter an integer value that is greater than `10` or less than `1`, or (ii) enters something that is not in proper integer format, like a floating-point number or text string.

Thus, six different runs of the program—the first three with correct input, and the others with incorrect input—could be:

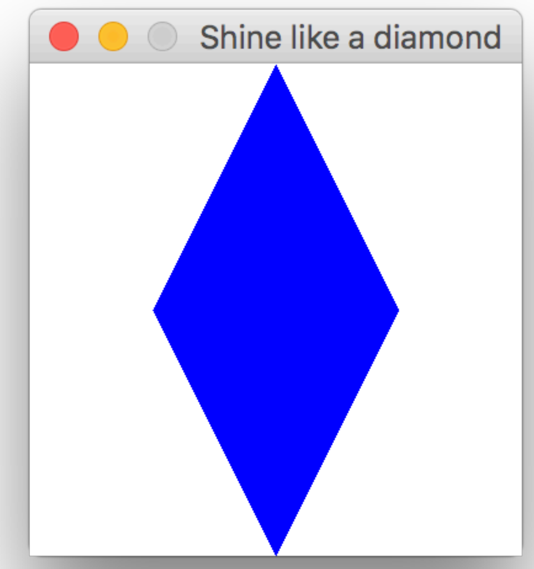
```
Please guess an integer between 1 and 10 (inclusive): 5
You guessed too high (correct answer: 4).

Please guess an integer between 1 and 10 (inclusive): 5
You guessed too low (correct answer: 6).

Please guess an integer between 1 and 10 (inclusive): 5
You guessed right!

Please guess an integer between 1 and 10 (inclusive): 21
That is an improper guess...

Please guess an integer between 1 and 10 (inclusive): 3.5
That is an improper guess...

Please guess an integer between 1 and 10 (inclusive): flapjacks
That is an improper guess...
```

```
// write the code for Question 5 here
import java.util.Scanner;

public class Q5
{
```

```
}
```

6. **(10 pts.)** **CODE COMPLETION (II).**

On the next page, complete the given class so that it can create something like the following image when run (although this is printed in black and white, here we are seeing a blue diamond, created out of two blue triangles).



a. Set the size of the window at random, so it is a square of side-length L, with $200 \leq L \leq 500$.

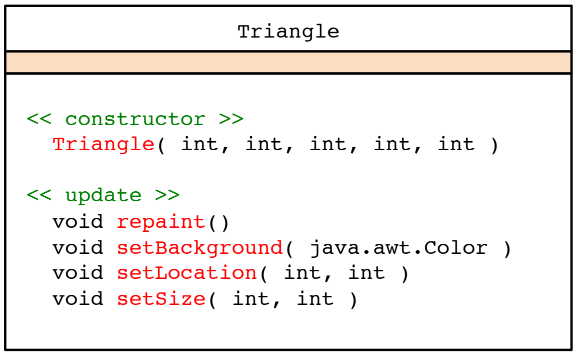
b. Create two `Triangle` objects; each should have height and width equal to ***one half*** of the size of the window; i.e., if the window itself is of size (300×300), then each of the shapes would be of size (150×150). They should be centered horizontally in the window, with one directly on top of the other, pointing up and down respectively.

c. Choose a random color for the two objects. Each time the program runs the two shapes should both be either red, blue or green, and each color should occur with equal probability on any given run of the program.

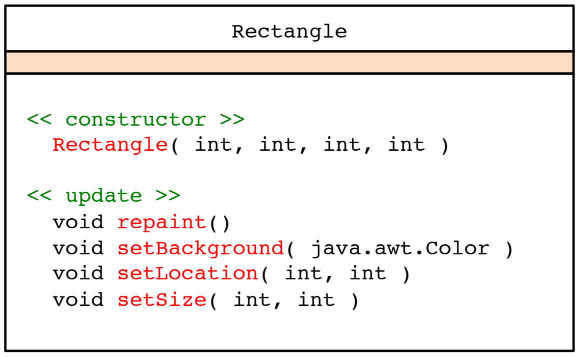
Note 1: Class diagrams for required graphical classes appear on the last page of the exam.

Note 2: The last input to the `Triangle()` constructor gives its orientation: if this number is `1`, then it will point upwards, and for any other value it will point downwards.

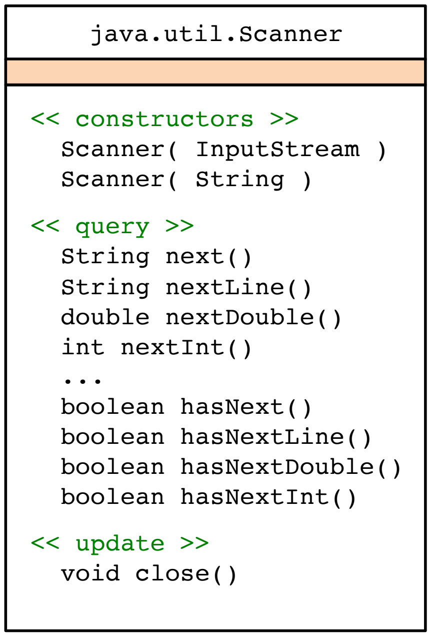
```
// Write the code for Question 6 here.
import java.awt.Color;

public class Q6
{
    public static void main( String[] args )
    {
        Window win = new Window();
        win.setTitle( "Shine On You Crazy Diamond" );
        win.setBackground( Color.white );









    }
}
```



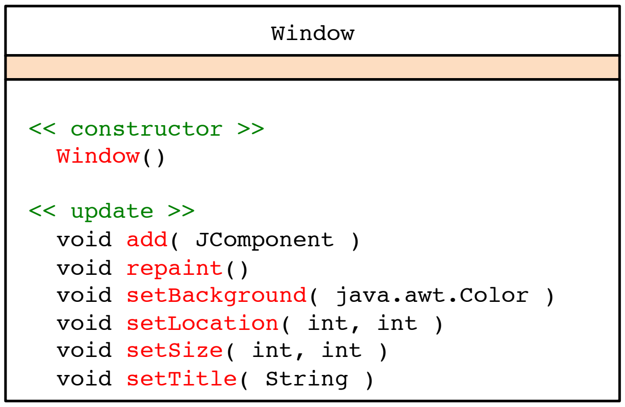
```
Triangle

<< constructor >>
  Triangle( int, int, int, int, int )

<< update >>
  void repaint()
  void setBackground( java.awt.Color )
  void setLocation( int, int )
  void setSize( int, int )
```



```
Rectangle

<< constructor >>
  Rectangle( int, int, int, int )

<< update >>
  void repaint()
  void setBackground( java.awt.Color )
  void setLocation( int, int )
  void setSize( int, int )
```



```
java.util.Scanner

<< constructors >>
  Scanner( InputStream )
  Scanner( String )

<< query >>
  String next()
  String nextLine()
  double nextDouble()
  int nextInt()
  ...
  boolean hasNext()
  boolean hasNextLine()
  boolean hasNextDouble()
  boolean hasNextInt()

<< update >>
  void close()
```



```
Window

<< constructor >>
  Window()

<< update >>
  void add( JComponent )
  void repaint()
  void setBackground( java.awt.Color )
  void setLocation( int, int )
  void setSize( int, int )
  void setTitle( String )
```