UNIVERSITY OF WISCONSIN-LA CROSSE
Department of Computer Science

CS 120 Software Design I Spring 2017
Practice Midterm Exam 01 Tuesday, 28 February 2017

- **Do not turn the page until instructed to do so.**
- **This booklet contains 9 pages including the cover page.**
- **This is a closed-book exam. All you need is the exam and a writing utensil.**
- **You have exactly 55 minutes.**
- **The maximum possible is 50.**

| PROBLEM | SCORE |
|---|---|
| 1 | 8 |
| 2 | 8 |
| 3 | 10 |
| 4 | 4 |
| 5 | 10 |
| 6 | 10 |
| TOTAL | 50 |

# ANSWER KEY

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.**
   (***Note**: I am including some explanations for a few of these; this is just for the purpose of the answer key, and you will not be expected to do the same on the actual exam.*)

   a. If you do not `import` a class for use in your code, then you cannot create or otherwise use objects of that class.

      **False**: you can also use the complete path address, or place the class code in a default location, such as the local source folder.

   b. Within a single code block, no object can have two different identifiers.

      **False**.

   c. Within a single code block, no two objects can be declared with the same identifier.

      **True**.

   d. Anywhere an object of class $C$ can go, a call to a non-`void` method that returns objects of type $C$ can also go.

      **True**.

   e. Variables can be named anything you like, without exception.

      **False**: not only must names be unique, but there are rules about what sorts of characters you can use (e.g., you cannot *start* the name with a number).

   f. The following line of code will not compile:

      ```
      int x = (int) 20 * 3.3;
      ```

      **True**: the `(int)` cast binds to just the `20`, and so the right-hand side is actually a `double` value, which will not auto-convert and lose precision.

   g. The following line of code will not compile:

      ```
      int x = 20 * (int) 3.3;
      ```

      **False**: the cast here binds to `3.3`, and so `x == 60` here.

   h. The following code will always print some result, *no matter what* value the integer `x` has:

      ```
      if( x < 0 )
          System.out.println( "One" );
      else if ( x != 0 )
          System.out.println( "Two" );
      ```

      **False**: the statement `if`-clause executes and prints `One` if the number is less than `0`, and the `else if`-clause executes and prints `Two` if the number is greater than `0`, but if it is *equal to* `0`, then nothing happens.

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

a. (*2 pts.*) Consider the class `Oval`, described on the last page of this exam. The `setBackground()` method takes a `java.awt.Color` object as **an input parameter** and has a **void return type**, which means that the method does not output a value when it is done.

b. (*2 pts.*) When we want to convert a more-precise primitive type to a less-precise type in Java, we must use **a cast command**. An example is: `int x = (int)( 10 / 3.5 * 2 );`

c. (*4 pts.*) If your code compiles correctly, this means it has correct Java syntax. However, errors can still occur when the code executes; the JVM will catch certain of these errors, called **runtime errors (or runtime exceptions)**. An example is **a null pointer exception**, which is caused when **our code tries to run a method using a variable that does not refer to any object in memory.**

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 = 3 / 2 * 4 + 6;` `x == 10`

b. `int x = (int)( 3 / 2.0 * 4 + 6 );` `x == 12`

c. `int x = 3 / (int)( 2.0 * 4 + 6 );` `x == 0`

d. `double x = 10 / 4 + 11;` `x == 13.0`

e. `double x = 10.0 / 4 + 11;` `x == 13.5`

f. `String x = "num = " + (3 + 6);` `x == "num = 9"`

g. `String x = "num = " + 3 + 6;` `x == "num = 36"`

h. `int x = (int)( Math.random() * 1 + 100);` `x == 100`

i. `int x = (int)( Math.random() * 100 ) + 1;` `1 <= x <= 100`

j. `boolean x = ( 2 != 2.0 );` `x == false`

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

Consider the following code:

---

```
Oval o1, o2, o3, o4;
o1 = new Oval( 50, 50, 100, 100 );
o2 = new Oval( 100, 100, 200, 200 );
o3 = null;
o4 = new Oval( 200, 200, 300, 300 );

o1 = o2;
o2 = o3;
o3 = o4;
o4 = o1;

o1.setBackground( Color.blue );
o2.setBackground( Color.red );
o3.setBackground( Color.green );
o4.setBackground( Color.magenta );
```

---

a. When this code is complete, two of the `Oval` variable identifiers refer to the same object in memory. What are those two identifiers?

**Answer:** `o4` and `o1`
(due to the last assignment: `o4 = o1;`).

b. This code orphans a single object in memory. Write down the code line that has this effect.

**Answer:** `o1 = o2;`
(this orphans the object to which `o1` originally referred).

c. This code will cause a `Null Pointer Exception` when it executes. Write down the code line that generates this error.

**Answer:** `o2.setBackground( Color.red );`
(the variable has a `null` reference that it gets from being assigned the same value as `o3`, which *started off* `null`, but isn't by the end of the code).

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

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

a. Asks the user for an integer value via `System.out`, and reads it in from `System.in`, using a `Scanner`.

b. Displays the **absolute value** of that input, so that if the user enters a negative number, it displays it in positive form. (See below for required format.)

c. Displays the **cube** of the value, so that if the user enters a number $n$, it will display the value of $n^3$.

d. Treats the required input value as zero if it is in incorrect form.

---

Thus, three different runs of the program—the first two with correct input, and the third with incorrect input–could be:

---

```
Please enter an integer value: -5
Absolute value: 5
Cube: -125


Please enter an integer value: 5
Absolute value: 5
Cube: 125


Please enter an integer value: banana
Absolute value: 0
Cube: 0
```

```
import java.util.Scanner;

public class Q5
{
    public static void main( String[] args )
    {
       Scanner scan = new Scanner( System.in );

       System.out.print( "Please enter an integer value: " );
       int i = 0;
       if ( scan.hasNextInt() )
           i = scan.nextInt();

        int abs = i;
        if ( abs < 0 )
            abs = -1 * abs;
        System.out.println( "Absolute value: " + abs );

        int cube = i * i * i;
        System.out.println ( "Cube: " + cube );

       scan.close();
    }
}
```

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

Complete the given class so that it can execute the following steps (use the back of the page if you run out of room):

a. Create two different random integer values that are either `1` or `2`.

b. If the first of the two number is **less than** the other, then a circle with diameter of 50 pixels is placed in the window, centered vertically and horizontally.

c. If the first of the two number is **greater than** the other, then a square with sides of 50 pixels is placed in the window, centered vertically and horizontally.

d. If the two numbers are the same, the background of the window is turned black.

**Note:** class diagrams for required graphical classes appear on the last page of the exam.

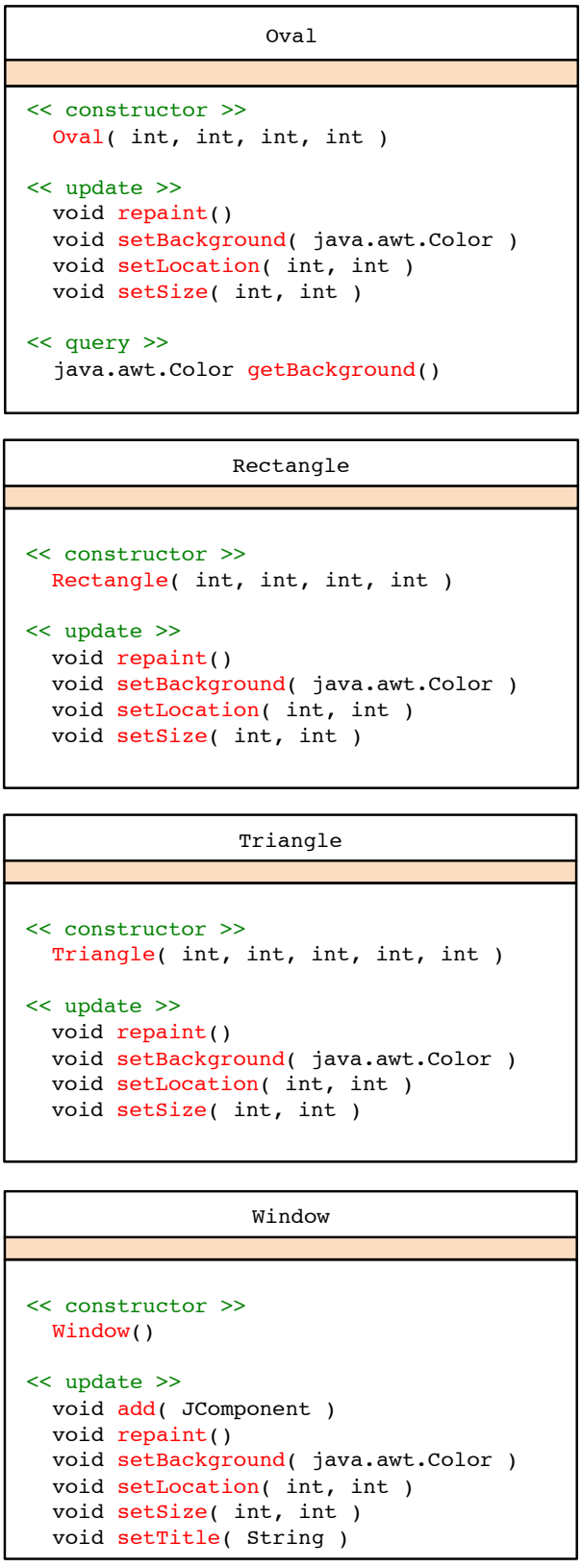
---

```
import java.awt.Color;

public class Q6
{
    public static void main( String[] args )
    {
        Window win = new Window();
        int winSize = 300;
        win.setSize( winSize, winSize );

        int val1 = (int)( Math.random() * 2 ) + 1;
        int val2 = (int)( Math.random() * 2 ) + 1;

        int shapeSize = 50;
        int shapeLoc = ( winSize / 2 ) - ( shapeSize / 2 );
        if ( val1 < val2 )
        {
            Oval o = new Oval( shapeLoc, shapeLoc, shapeSize, shapeSize );
            win.add( o );
        }
        else if ( val1 > val2 )
        {
            Rectangle r = new Rectangle( shapeLoc, shapeLoc, shapeSize, shapeSize );
            win.add( r );
        }
        else
            win.setBackground( Color.black );
   }
}
```


Oval
<< constructor >>
Oval( int, int, int, int )
<< update >>
void repaint()
void setBackground( java.awt.Color )
void setLocation( int, int )
void setSize( int, int )
<< query >>
java.awt.Color getBackground()
Rectangle
<< constructor >>
Rectangle( int, int, int, int )
<< update >>
void repaint()
void setBackground( java.awt.Color )
void setLocation( int, int )
void setSize( int, int )
Triangle
<< constructor >>
Triangle( int, int, int, int, int )
<< update >>
void repaint()
void setBackground( java.awt.Color )
void setLocation( int, int )
void setSize( int, int )
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 )