UNIVERSITY OF WISCONSIN-LA CROSSE
Department of Computer Science

CS 120 **Software Design I** **Spring 2017**
Final Exam **11 May 2017**

- **Do not turn the page until instructed to do so.**
- **This booklet contains 11 pages including the cover page.**
- **This is a closed-book exam. All you need is the exam and a writing utensil. (A calculator is permitted.)**
- **You have exactly 120 minutes.**
- **The maximum possible score is 80 points.**

PROBLEM	SCORE
1	
2	
3	
4	
5	
6	
7	
8	
9	
TOTAL	

NAME: _____

1. **(10 pts.) TRUE/FALSE.**

   Label each statement true or false. **You do not need to explain your answers.**

   a. The following contains an error, and will not compile:

   ```
   String s = 3 + 2;
   System.out.println( s );
   ```

   b. The following contains an error, and will not compile:

   ```
   System.out.println( 3 + 2 );
   ```

   c. The following loop runs exactly 5 times:

   ```
   for ( int i = 1; i <= 10; i = i + 2 )
       System.out.print( i + " " );
   ```

   d. The following loop runs exactly 5 times:

   ```
   for ( int i = 0; i <= 10; i = i + 2 )
       System.out.print( i + " " );
   ```

   e. Any variable declared inside a loop is always local to that loop, and cannot be accessed outside the loop.

   f. After running the following code, the variable `length == 0`.

   ```
   String s;
   int length = s.length();
   ```

   g. After running the following code, the array element `arr[0] == 0`.

   ```
   int[] arr = new int[10];
   ```

   h. After running the following code, the array element `arr[0] == ""` (an empty `String`).

   ```
   String[] arr = new String[10];
   ```

   i. A Java class can only `implement` a single interface at one time.

   j. Everybody likes a free point on an exam!

2. **(10 pts.) SHORT ANSWER.**

i. (5 pts.) To over-ride a method, the original version and the new version ***must have*** 4 things in common:

i. _____

ii. _____

iii. _____

iv. _____

One thing that ***can be different*** between the two versions of the method is:

v. _____.

ii. (3 pts.) What `String` method can cause `StringIndexOutOfBoundException` errors?

These errors are caused by improper input values to the method. What are two different ways that these input values can be incorrect?

iii. (2 pts.) Suppose we have a class, `Driver`, with code using two arrays and an object of type `Converter`:

```
String[][] strs = new String[5][10];
Converter con = new Converter();
String[] output = con.convert( strs, "Test" );
```

Without knowing what the `convert()` method does, we do know what its method signature (i.e., its first line) must be. What will it look like, exactly?

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

For each of the following, give the output of the code.

```
int base = 2;
while( base < 6 )
{
    int pow = base;
    for( int e = 2; e != 4; e++ )
    {
        pow = pow * pow;
        System.out.println( pow + ", " + e );
    }
    base = base + 3;
}
```

```
int[] arr = { 5, 2, 1, 6 };
int[] sums = new int[arr.length];
for ( int i = 0; i < arr.length; i++ )
{
    int s = 0;
    for ( int j = 0; j < arr.length; j++ )
    {
        if ( arr[j] >= arr[i] )
        {
            s = s + arr[j];
        }
    }
    sums[i] = s;
    System.out.println( arr[i] + ": " + sums[i] );
}
```

4. **(5 pts.) CODE EVALUATION, II.**

Suppose we have the following two class definitions:

```
public class ClassA {
    protected int x;

    public ClassA( int xin ) {
        x = xin;
    }

    public int getValue() {
        return x;
    }
}

public class ClassB extends ClassA {
    private int y;

    public ClassB( int z, int yin ) {
        super(z);
        y = yin;
    }

    public int getY() {
        return y;
    }

    public int getValue() {
        return x + y;
    }
}
```

Suppose now we create two objects as follows:

```
ClassA a = new ClassA( 1 );
ClassB b = new ClassB( 2, 3 );
```

Circle the line(s) among the two immediately below that will **not** compile properly, based on the code above that creates the variables `a` and `b`.

```
int u = a.getY();

int v = b.getY();
```

After the following, what are the values of the variables `i1`, `i2`, and `i3`, based on the code above that creates the variables `a` and `b`? (Write **ERROR** for any value if that line of code will not compile properly.)

```
int i1 = a.getValue();
int i2 = b.getValue();
int i3 = b.getY();
```

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

Write a method called `sumPositive()` below. This method should:

- Take a 2-dimensional array of integers as input. This array ***can be ragged*** (with different numbers of columns per row).
- Return an integer array; each element of the array should consist of the sum of values that are greater than 0 in the corresponding input array row.

Thus, if we have the following code:

```
int[][] nums = { { -1, 2, 3 }, { 2, -4, 3, 1 }, { -3, -4 } };
int[] out = sumPositive( nums );
```

then when we are done, we will have the array `out == { 5, 6, 0 }`.

For full points, your code must use **nested loops**, each of which is actually used to generate the output. (You may use whatever types of loops you choose.)

---

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

Fill in the class below. It should have two methods.

- The first, `countDollars()`, should take in a `String` as input, and return the number of dollar signs (`$`) in it.
- The second, `highCount()`, should take in two `String` objects and a single character as input, and return the string that has the highest number of occurrences of that character. Thus, calling `highCount( "test", "tattle", 't' )` would return `"tattle"`, since it has three occurrences of `'t'`, whereas `"test"` has only two. In the event of a tie, it should return the ***first*** input string, so `highCount( "test", "tutor", 't' )` would return `"test"`.

```
public class Question6
{









}
```

7. **(10 pts.) CODE COMPLETION, III.**

Add the `shuffle()` method to the class below:

- If the input arrays are of *equal* length, it should return a new array containing all the elements of each in *alternating order*: the first element of the first input, then the first element of the second, followed by the second element of the first input and second element of the second input, and so on. Thus, after the first call below, the output array `out1` would look like:

  `{ "Ace", "10", "King", "9", "Queen", "8", "Jack", "7" }`

- If the arrays are of *different* lengths, then it will simply return an *empty array*. Thus, after the second call below, output array `out2` will be empty.

```
public class Question7
{
     public Question7()
     {
        String[] hand1 = { "Ace", "King", "Queen", "Jack" };
        String[] hand2 = { "10", "9", "8", "7" };
        String[] hand3 = { "2", "3" };

        String[] out1 = shuffle( hand1, hand2 );
        String[] out2 = shuffle( hand1, hand3 );
     }









}
```

8. **(10 pts.) CODE COMPLETION, IV.**

The class below uses another class called `TripMileage`, which you will write, as follows:

- The class will contain two instance variables to store current values for distance driven and amount of gas used; these will be stored as `double` values.
- It will contain a `void` method called `setMileage()` that takes in two `double` values, one to be assigned to the distance variable, and one to be assigned to the gas variable.

  The method should check its input parameters. A valid parameter for `distance` must be ***greater than or equal to*** `0`, and a valid parameter for `gas` must be ***greater than*** `0`. If ***both*** parameters are valid, then the method should assign these values to the proper instance variables; otherwise the input values should be ignored, and the currently saved values will not change.
- It will contain two methods, `getDistance()` and `getGas()`, that return the saved values.

If written correctly, the code below will produce output as follows; note that in the last two lines of output, the saved values do not change, since the inputs to `setMileage()` are invalid.

```
Distance: 0.0, Gas = 5.0
Distance: 10.0, Gas = 10.0
Distance: 10.0, Gas = 10.0
Distance: 10.0, Gas = 10.0
```

```
public class Question8
{
   public static void main( String[] args )
   {
      TripMileage tm = new TripMileage();

      tm.setMileage(0, 5);
      System.out.println("Distance: " + tm.getDistance() + ", Gas = " + tm.getGas());

      tm.setMileage( 10, 10 );
      System.out.println("Distance: " + tm.getDistance() + ", Gas = " + tm.getGas());

      tm.setMileage( -3, 5 );
      System.out.println("Distance: " + tm.getDistance() + ", Gas = " + tm.getGas());

      tm.setMileage( 5, 0 );
      System.out.println("Distance: " + tm.getDistance() + ", Gas = " + tm.getGas());
   }
}
```

```
// Write the TripMileage class here.
```

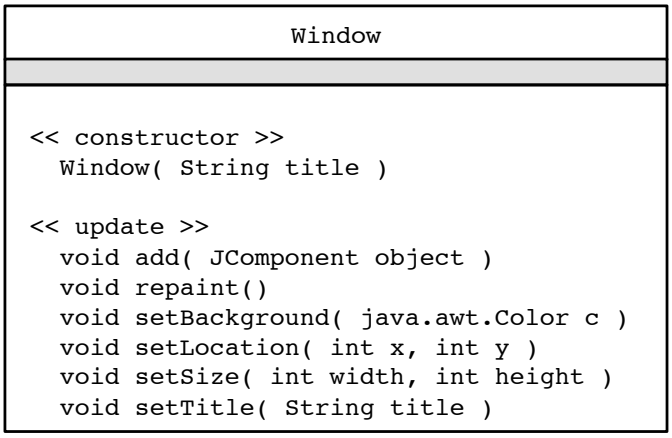
9. **(5 pts.) CODE COMPLETION, V.**

At the bottom of this page, write a class, called `BetterWindow`. Your code should function as follows:

(a) The `BetterWindow` class should extend the `Window` class, which has the usual methods for changing size, location, etc., shown in the below class diagram. It has a constructor that takes a title as input; calling that constructor will create a window with the given title displayed at the top.

(b) The `BetterWindow` constructor should also take a title as input and display it entirely in capital letters. The window should always start off with width and height both 250. Thus, the following creates a (250×250) window with title `"MY WINDOW"`:

```
BetterWindow win = new BetterWindow( "My window" );
```



```
// Write the BetterWindow class here and on the back of the page if necessary.
```