FINAL EXAM

CMPS 12a - Spring 02

Thomas Raffill

Name:_____

Student ID:_____

This exam is closed book, closed notes, no electronic devices. Show all work. Partial credit given for partial solutions. Presentation counts! Be legible and coherent for full credit.

Question 1:_____(out of 5)
Question 2:_____(out of 5)
Question 3:_____(out of 5)
Question 4:_____(out of 5)
Question 5:_____(out of 5)
Question 6:_____(out of 5)
Question 7:_____(out of 5)
Question 8:_____(out of 5)
Question 9:_____(out of 5)
Question 10:_____(out of 5)
Question 11:_____(out of 5)
Question 12:_____(out of 5)
Question 13:_____(out of 10)
Question 14:_____(out of 10)
Question 15:_____(out of 10)
Question 16:_____(out of 20)
Question 17:_____(out of 10)

Total:_____(out of 120)
(Anything above 100 counts for extra credit)

Name:_____

TRUE OR FALSE: Give the best answer, true or false, to the following statements.

1. (5 points) If you spend time early making a good design, you can save time overall because developing the program will be much easier.
   Solution: True. In programming, most of the time goes into figuring out the right thing to do. So it is better to do as much of this as possible in the beginning so you won't waste time developing incorrect code.

2. (5 points) It is a smart idea to omit comments in your code if you want to compile to a smaller class file.
   Solution: False. The compiler will ignore comments. Therefore, adding or taking away comments will not change the size of the class file. It is not a smart idea to omit comments because it makes code less readable.

3. (5 points) When writing a class, you should make all of the instance variables public so you won't need to write get and set methods for them.
   Solution: False. It is generally bad style to make all the instance variables public. It is dangerous because it will allow other classes to modify instances of your class in unexpected ways.

Name:_____

MULTIPLE CHOICE: Pick the best answer from the following choices.

4. (5 points) Which boolean expression is equivalent to the expression:

```
!(x % 3 != 0 || !(x % 5 == 0 && x % 7 == 0))
```

HINT: If confused, try plugging in small numbers for x.

(a)
```
(x % 3 == 0 && x % 35 != 0)
```

(b)
```
(x % 3 == 0 && (x % 5 != 0 || x % 7 != 0))
```

(c)
```
(x % (3*5*7) == 0)
```

Solution: (c). We apply the laws of boolean algebra:

```
!(x % 3 != 0 || !(x % 5 == 0 && x % 7 == 0))
is equivalent to
!(x % 3 != 0) && !(!(x % 5 == 0 && x % 7 == 0))
is equivalent to
(x % 3 == 0) && (x % 5 == 0) && (x % 7 == 0)
```

Now we apply simple precalculus mathematics. If a number is divisible by 3 and by 5 and by 7, then it must be divisible by $3 \cdot 5 \cdot 7$. So the last boolean expression is equivalent to:

```
(x % 3*5*7 == 0)
```

You could also arrive at the answer by testing with small values of x. When x is a multiple of 3 but not a multiple of 5 or 7, expressions (a) and (b) will both become true but the expression of the question and expression (c) will become false. You would have disovered this if you tested with x = 0, x = 3, x = 6, etc.You might also note that expressions (a) and (b) are equivalent to each other (because a number is not a multiple of $5 \cdot 7 = 35$ if it is not a multiple of either 5 or 7).

Name:_____

5. (5 points) Which of the following is a valid Java identifier?

   (a) `1More4TheRoad`

   (b) `double`

   (c) `$500`

   Solution: (c). A Java identifier cannot begin with a number, so (a) is not valid. A Java identifier cannot be a reserved word like the keyword `double`, so (b) is not valid. The $ symbol is a Java letter, so $500, which begins with a Java letter and consists of Java letters and numbers, is valid.

6. (5 points) Which of the following is a valid Java literal?

   (a) `"A+B+C"`

   (b) `'A'+'B'+'C'`

   (c) `"A"+"B"+"C"`

   Solution: (a). (a) is a String literal, but (b) is an expression of the sum of three char literals, and (c) is an expression of the concatenation of three String literals.

Name:_____

EVALUATING EXPRESSIONS

Give the values of the following expressions.

7. (5 points)

```
(char)(('m' + 4)/('z' - 'y'))
```

Solution: 'q'. Obviously, 'z' - 'y' equals 1 because they are one character apart alphabetically, and dividing by 1 changes nothing. The expression 'm' + 4 is widened to an int value, but we cast the result back to a char. Therefore, the result is four letters past 'm', which is 'q'.

8. (5 points) (Assume a is an int with a value of 2)

```
(double)(a++/4)+(double)a/4
```

Solution: 0.75. Applying the rules of precedence, the parenthesization of the expression is as follows:

```
((double)((a++)/4))+(((double)a)/4)
```

We first increment a. Since it is postfix increment, this returns a value of 2 but changes a's value to 3. We then perform the division 2/4. Since both are int literals, we use integer division, and this returns a value of 0. Next, we cast this result to a double. This gives 0.0. Then we add this to the right expression. On the right side we first cast a to a double, giving 3.0. Then we divide by 4, which gives 0.75. Finally, 0.0 + 0.75 gives the result of 0.75.

9. (5 points) (Assume s has already been declared as a String variable)

```
(s = "555").equals("" + 5 + 5 + 5)
```

Solution: true. The subexpression (s = "555") returns a String whose value is "555". The subexpression "" + 5 + 5 + 5 is a concatenation of the empty String with three 5's, which also gives a String whose value is "555". When we call the equals method on the first String with the second String, it returns true because the two Strings are the same.

Name:_____

RECOGNIZING EXCEPTIONS

Identify the types of Exceptions in the following code fragments. The exceptions will be from the following list:

(a) NullPointerException

(b) ArrayIndexOutOfBoundsException

(c) ArithmeticException

10. (5 points)

```
int[] smallPrimes = { 2, 3, 5, 7, 11, 13, 17, 19};
int x = 5/(smallPrimes[2] - smallPrimes[1] - (smallPrimes[7] - smallPrimes[6]));
```

Solution: ArithmeticException. When we access the array elements, the expression becomes $5/(5-3-(19-17)) = 5/(2-2) = 5/0$, and this throws an ArithmeticException because division by 0 is not allowed.

11. (5 points)

```
int[] squares = { 1, 4, 9, 16, 25 };
int x = 5/(squares[5] - (squares[3] + squares[4]));
```

Solution: ArrayIndexOutOfBoundsException. The highest index in this array is 4, so squares[5] is out of bounds.

12. (5 points)

```
Random r;
r.nextInt(10);
```

Solution: NullPointerException. We attempt to reference the variable r before allocating it with new.

Name:_____

13. (10 points)
Space explorers have discovered the intergalactic life forms who have a different measurement system for stellar magnitude. The conversion formula is given by $I = (3/5)H + 1001$, where H is a magnitude in the human scale and I is a magnitude in the intergalactic scale. Write a public static function called humanToIntergalactic taking a double parameter and returning a double value and implementing the above conversion formula.
Solution:

```
public static double humanToIntergalactic(double H) {
  return (double)3/5 * H + 1001;
}
```

It is important to cast to double or use double literals, because otherwise 3/5 will be interpreted as integer division and return a value of 0.

14. (10 points)
You have a 3-dimensional array of double values called **spaceMagnitudes** giving the brightness of points in a certain sector in interstellar space given in the human stellar magnitude scale. Write a code fragment to step through this array and convert each element from the human magnitude scale to the intergalictic magnitude scale using the humanToIntergalactic function you defined in the previous problem.
Solution:

```
for (int i = 0; i < spaceMagnitues.length; i++) {
  for (int j = 0; j < spaceMagnitues[i].length; j++) {
    for (int k = 0; k < spaceMagnitues[i][j].length; k++) {
      spaceMagnitudes[i][j][k] = humanToIntergalactic(spaceMagnitudes[i][j][k]);
    }
  }
}
```

Name:_____

15. (10 points)
What does the following code print out?

```
public class Cities {
  private String myCity = "Santa Cruz";

  public static void main(String[] args) {
    Cities myCity = new Cities();
    String result = myCity.addState("California");
    System.out.println("myCity is " + myCity.myCity);
    System.out.println("The result is " + result);
    }

  public String addState(String state) {
    myCity = myCity + ", " + state;
    return "city: " + myCity + " state: " + state;
    }
  }
```

Solution:

```
myCity is Santa Cruz, California
The result is city: Santa Cruz, California state: California
```

There are two variables called myCity: the String instance variable and the Cities variable in main. The myCities instance variable is initialized to "Santa Cruz" by the constructor. Then the addState method concatenates ", California" to it. It returns the String
"city: Santa Cruz, California state: California".

Name:_____

16. (20 points)
Write a class called **Angle** with one private int instance variable called **degrees**, a public constructor taking one int parameter, public **getDegrees** and **setDegrees** methods, a static **add** method taking two Angles and returning an Angle representing their sum, and an instance **addTo** method taking one Angle and adding it to the given instance.

Your class should enforce the constraint that the **degrees** variable is always a number between 0 and 359 by suitable use of the remainder operation in the constructor, setDegrees, add and addTo methods.
Solution: We use the modulo operator to enforce the constraint.

```
public class Angle {
  private int degrees

  public Angle(int d) {
    degrees = d % 360;
  }

  public int getDegrees() {
    return degrees;
  }

  public void setDegrees(int d) {
    degrees = d % 360;
  }

  public static Angle add(Angle a, Angle b) {
    return new Angle((a.degrees + b.degrees) % 360);
  }

  public void addTo(Angle a) {
    degrees = (degrees + a.degrees) % 360;
  }
}
```

Name:_____

17. (10 points)
You have a 2-dimensional array of String values called stringArray. Write a code fragment to copy the elements of the array into a new array called transposedStringArray which holds the same contents as the original array but with the indices transposed. That is, if stringArray[i][j] contains some String, then transposedStringArray[j][i] should contain the same String.
Solution: We allocate the transposed array with the transposed size of the original array (the number of rows in the transposed array is the number of columns in the original array, and vice versa). Then we step through the array and copy the i,j-th element of the original array to j,i-th element of the transposed array.

```
String[][] transposedStringArray
  = new String[stringArray[0].length][StringArray.length];
for (int i = 0; i < stringArray.length; i++) {
  for (int j = 0; j < stringArray[i].length; j++) {
    transposedStringArray[j][i] = stringArray[i][j];
  }
}
```