Signature ____________________ **Name** ________________________

cs11f ____ **Student ID** ____________________

CSE 11 Midterm Fall 2011

Page 1 ___________ (18 points)

Page 2 ___________ (18 points)

Page 3 ___________ (31 points)

Page 4 ___________ (13 points)

Page 5 ___________ (8 points)

Total ___________ (88 points = 84 base points + 4 points EC [5%])
(84 points = 100%)

This exam is to be taken by yourself with closed books, closed notes, no electronic devices.
You are allowed one side of an 8.5"x11" sheet of paper handwritten by you.

(Partial) Operator Precedence Table

Operators				Associativity
!	++	-- (pre & post inc/dec)		right to left
*	/	%		left to right
+	-			left to right
<	<=	>	>=	left to right
==	!=			left to right
&&				left to right
||				left to right
=				right to left

1) What are the values of the indicated variables after the following code segments are executed?

```
int x = 5, y = 3, z;
boolean bool1 = !((x > 4) || (y <= 6)) == ((y <= 4) && !(x > 6));

if ( x++ >= 4 || --y >= 3 )
  z = x++ + --y;
else
  z = ++x + y--;
```

bool1 =
x =
y =
z =

```
int a = 5, b = 3, c;
boolean bool2 = !(b > 4) && (a <= 6) && (a <= 4) || (b > 6);

if ( a++ >= 4 && --b >= 3 )
  c = a++ + --b;
else
  c = ++a + b--;
```

bool2 =
a =
b =
c =

What gets printed?

```
public class While
{
  public static void main( String[] args )
  {
    final int MAX = 9, MIN = 5;
    int i = 7, j = 8;

    while ( i <= MAX )
    {
      while ( j > MIN )
      {
        ++j;
        System.out.println( i + " " + j );
        j -= 4;
      }
      i++;
      j = i;
    }

    System.out.println( i + " " + j );
  }
}
```

2) Which of the following <u>are</u> valid Java identifiers? (Circle your answer(s).)

```
1stJavaClass      My-First-Java-Class     sEvEnTeEn        CSE11Is_1_

CSE11Is#1         CSE_11                  My1stJavaClass   float
```

Given the following definition of class Thing2, what is the output of the Java application Test2?

```
class Thing2
{
  private int count;

  public Thing2( int count )
  {
    this.count = count;
  }

  public int getCount()
  {
    return this.count;
  }

  public void setCount( int count )
  {
    this.count = count;
  }

  public String toString()
  {
    String s = " ";

    switch( this.count )
    {
      case 3:
        s = s + "tres ";
        break;

      case 2:
        s = s + "duo ";

      case 1:
        s = s + "uno ";
        break;

      default:
        s = s + "mucho ";
        break;
    }

    return s;
  }

  public void swap1( Thing2 t2 )
  {
    Thing2 temp;
    Thing2 t1 = this;

    temp = t1;
    t1 = t2;
    t2 = temp;
  }

  public void swap2( Thing2 t2 )
  {
    int temp;

    temp = this.getCount();
    this.setCount( t2.getCount() );
    t2.setCount( temp );
  }
}
```

```
public class Test2
{
  public static void main( String[] args )
  {
    Thing2 first = new Thing2( 4 );
    Thing2 second = new Thing2( 2 );

    Thing2 temp = first;
    first = second;
    second = temp;

    System.out.println( first.toString() );
    System.out.println( second.toString() );

    Thing2 third = new Thing2( 1 );
    Thing2 fourth = new Thing2( 3 );

    third.swap2( fourth );

    System.out.println( third.toString() );
    System.out.println( fourth.toString() );

    first.setCount( third.getCount() );
    fourth = second;

    System.out.println( first == third );
    System.out.println( second == fourth );
    System.out.println( first.toString().equals( third.toString() ) );
    System.out.println( second.toString().equals( fourth.toString() ) );

    System.out.println( first.toString() );
    System.out.println( second.toString() );
    System.out.println( third.toString() );
    System.out.println( fourth.toString() );

    first = new Thing2( 5 );
    second = new Thing2( 2 );

    first.swap1( second );

    System.out.println( first.toString() );
    System.out.println( second.toString() );
  }
}
```

<u>Output</u>

____

____

____

____

____

____

____

____

____

____

____

____

____

____

3) What output is produced by the following program?

```
 1  public class Test3
 2  {
 3    private int a;
 4    private static int b = 2;
 5    private int c;

 6    public static void main( String[] args )
 7    {
 8      Test3 ref = new Test3( 3 );

 9      ref.method1( ref.a );
10    }

11    public Test3( int a )
12    {
13      this.a = a;
14    }

15    public void method1( int x )
16    {
17      int c = x--;
18      int b;

19      b = a + 2;
20      a = c + 3;

21      System.out.println( "this.a = " + this.a );
22      System.out.println( "Test3.b = " + Test3.b );
23      System.out.println( "this.c = " + this.c );
24      System.out.println( "c = " + c );
25      System.out.println( "b = " + b );
26      System.out.println( "a = " + a );
27      System.out.println( "result = " + method2( b + c ) );
28      System.out.println( "this.a = " + this.a );
29      System.out.println( "Test3.b = " + Test3.b );
30      System.out.println( "this.c = " + this.c );
31      System.out.println( "x = " + x );
32      System.out.println( "a = " + a );
33      System.out.println( "b = " + b );
34      System.out.println( "c = " + c );
35    }

36    private int method2( int x )
37    {
38      int b = x;
39      int c = this.c + Test3.b;

40      x = a = b + c;

41      System.out.println( "this.a = " + this.a );
42      System.out.println( "Test3.b = " + Test3.b );
43      System.out.println( "this.c = " + this.c );
44      System.out.println( "x = " + x );
45      System.out.println( "a = " + a );
46      System.out.println( "b = " + b );
47      System.out.println( "c = " + c );

48      Test3.b = b + 2;
49      this.c = a + c;

50      return x + 5;
51    }
52  }
```

Use the numbers below to identify various program parts.

1) static method
2) constructor
3) class definition (type)
4) instance method
5) local variable
6) static variable
7) instance variable
8) formal parameter
9) actual argument

_____ `method1()` on line 15   _____ `b` on line 18

_____ `Test3()` on line 11   _____ `a` on line 3

_____ `b + c` on line 27   _____ `a` on line 11

_____ `main()` on line 6   _____ `ref` on line 8

_____ `Test3` on line 1   _____ `b` on line 4

Output

```
this.a = ________
Test3.b = ________
this.c = ________
c = ________
b = ________
a = ________
this.a = ________
Test3.b = ________
this.c = ________
x = ________
a = ________
b = ________
c = ________
result = ________
this.a = ________
Test3.b = ________
this.c = ________
x = ________
a = ________
b = ________
c = ________
```

4)

What gets printed if the value of the actual argument passed to this method is 0? _______

```
public void f5( int x )
{
   int y = 0;

   if ( x <= 1 )
      y = 3;
   if ( x <= 2 )
      y = 5;
   if ( x == 3 || x >= 4 )
      y = 7;
   else
      y = 9;

   System.out.println( y );
}
```

What gets printed if the value of the actual argument passed to this method is 0? _______

```
public void f5( int x )
{
   int y = 0;

   if ( x <= 1 )
      y = 3;
   else if ( x <= 2 )
      y = 5;
   else if ( x == 3 || x >= 4 )
      y = 7;
   else
      y = 9;

   System.out.println( y );
}
```

What is the output of this recursive method if it is invoked as `ref.mystery( 10 );`? Draw Stack Frames to help you answer this question.

```
int mystery( int a )
{
  int b = a - 2;

  if ( b >= 7 )
  {
    System.out.println( a + " " + b );
    a = b - mystery( b + 1 );
  }
  else
  {
    System.out.println( "Stop" );
    b = a + 2;
  }

  System.out.println( a + " " + b );
  return a + b;
}
```

<u>Output</u>

5) Given the following definitions:

```
public interface Doable
{
  public abstract String doit();
}
```

```
public class Thing1 implements Doable
{
  private String str;

  public Thing1()
  {
    this.str = "Me";
  }

  public String speak()
  {
    return this.str;
  }

  public String doit()
  {
    return "Thing1 did it!";
  }
}
```

```
public class Thing2 implements Doable
{
  private String str;

  public Thing2()
  {
    this.str = "No, Me";
  }

  public String speak( String s )
  {
    return this.str + s;
  }

  public String doit()
  {
    return "Thing2 does it too!";
  }
}
```

And the following variable definitions:

```
Thing1 thing1 = new Thing1();
Thing2 thing2 = new Thing2();
Doable doable;
```

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error".

```
doable = thing1;

System.out.println( doable.getClass().getName() ); ____________________________

System.out.println( doable.doit() );               ____________________________

System.out.println( thing1.speak() );              ____________________________

doable = thing2;

System.out.println( doable.getClass().getName() ); ____________________________

System.out.println( doable.doit() );               ____________________________

System.out.println( thing2.speak( " Here" ) );     ____________________________
```

What two changes/additions would be needed to the above interface and class definitions so `doable.speak()` would compile and run for all valid assignments to `doable`? Be specific what needs to be added to which file(s). Do not remove or change any of the existing code.

1)

2)

Scratch Paper