Signature _____ **Name** _____

cs11f ____ **Student ID** _____

By filling in the above and signing my name, I confirm I will complete this exam with the utmost integrity and in accordance with the Policy on Integrity of Scholarship.

CSE 11
Final
Fall 2012

Page 1 _____ (17 points)

Page 2 _____ (26 points)

Page 3 _____ (24 points)

Page 4 _____ (19 points)

Page 5 _____ (10 points)

Page 6 _____ (19 points)

Page 7 _____ (20 points)

Page 8 _____ (12 points)

Page 9 _____ (35 points)

Page 10 _____ (11 points)

Total _____ (193 points = 183 base points + 10 points EC)
(100%) [>5%]

This exam is to be taken by yourself with closed books, closed notes, no electronic devices.
You are allowed both sides 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) Which of the following <u>are not</u> valid Java identifiers? (Circle your answer(s).)

[+1 correct; -1 incorrect; no neg score]

```
quarter_.25        Half-Time              25Cents            double

quarter25          SuperBowl_XLVII        QUARTER_25         string_cheese
```

2) Using the operator precedence table above, evaluate each expression and state what gets printed.

```
int a = 7, b = -1, c = 4;

System.out.println( a / c + a % c );                          _________

System.out.println( !(a > 4 || b <= 6) == a >= 4 || c < 6 );  _________

System.out.println( !( a + b <= c ) );                        _________

System.out.println( a + b * c - c / a % c );                  _________
```

3) What are the values of the indicated variables after the following code segments are executed? Remember short-circuit evaluation with && and ||.

```
int a = 6, b = 8, c;
boolean bool1 = !(b > 4) && (a <= 6) && (a <= 4) || !(b > 6);

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

bool1 =
a =
b =
c =

```
int x = 6, y = 8, z;
boolean bool2 = !((x > 4) || (y <= 6)) != ((y <= 4) && (x > 6));

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

bool2 =
x =
y =
z =

What is the equivalent Java expression for the following expression such that no ! operators are used?

```
!( x < -10 || y >= 7 )
```

_____

4) What gets printed?

```
public class Question4
{
  public static void main( String[] args )
  {
    final int MAX = 7, MIN = 2;
    int i = 5, j = 3;

    while ( i >= MIN )
    {
      while ( j < MAX )
      {
        ++j;
        System.out.println( i + " " + j );
        j += 2;
      }
      j = i;
      i--;
    }
    System.out.println( i + " " + j );
  }
}
```

5) What gets printed as a result of the call f5(1, 3)? _____

```
public static void f5( int a, int b )
{
  if ( (a > 0) && (b > 0) )
  {
    if ( a > b )
    {
      System.out.println( "A" );
    }
    else
    {
      System.out.println( "B" );
    }
  }
  else if ( (a < 0) || (b < 0) )
  {
    System.out.println( "C" );
  }
  else
  {
    System.out.println( "D" );
  }
}
```

Give an example of values passed as arguments to f5() that would result in the method printing "D".

f5(_____ , _____);

6) Assume a program had the following definitions (a Point has an x and a y value):

```
Point p1 = new Point( 420, 42 );
Point p2 = new Point( p1 );
Point p3 = p1;
```

What results would be produced by evaluating the following expressions?

```
p1 == p2      _____________        p1 == p3      _____________        p2 == p3      _____________

p1.equals(p2) _____________        p1.equals(p3) _____________        p2.equals(p3) _____________

p3.translate(1, 1);  // Add 1 to the x and y coordinates in the Point object ref'ed by p3

p1.equals(p2) _____________        p1.equals(p3) _____________        p2.equals(p3) _____________
```

7) Assume we have a Java source file with the source code for a class named `public class Boogie` which requires the `objectdraw.jar` library (located in the same/current directory) to compile and run.

Write the full Unix command to compile this Java program: _____

This command will produce a file named: _____

Write the full Unix command to run this as a Java application: _____

8) What is the output of the following program?

```
public class Tricky
{
  public static void main( String[] args )
  {
    foo1();
    System.out.println( "main1" );
    foo2();
    System.out.println( "main2" );
    foo3();
    System.out.println( "main3" );
  }

  public static void foo1()
  {
    System.out.println( "A" );
  }

  public static void foo2()
  {
    System.out.println( "B" );
    foo3();
    System.out.println( "C" );
  }

  public static void foo3()
  {
    System.out.println( "D" );
    foo1();
    System.out.println( "E" );
  }
}
```

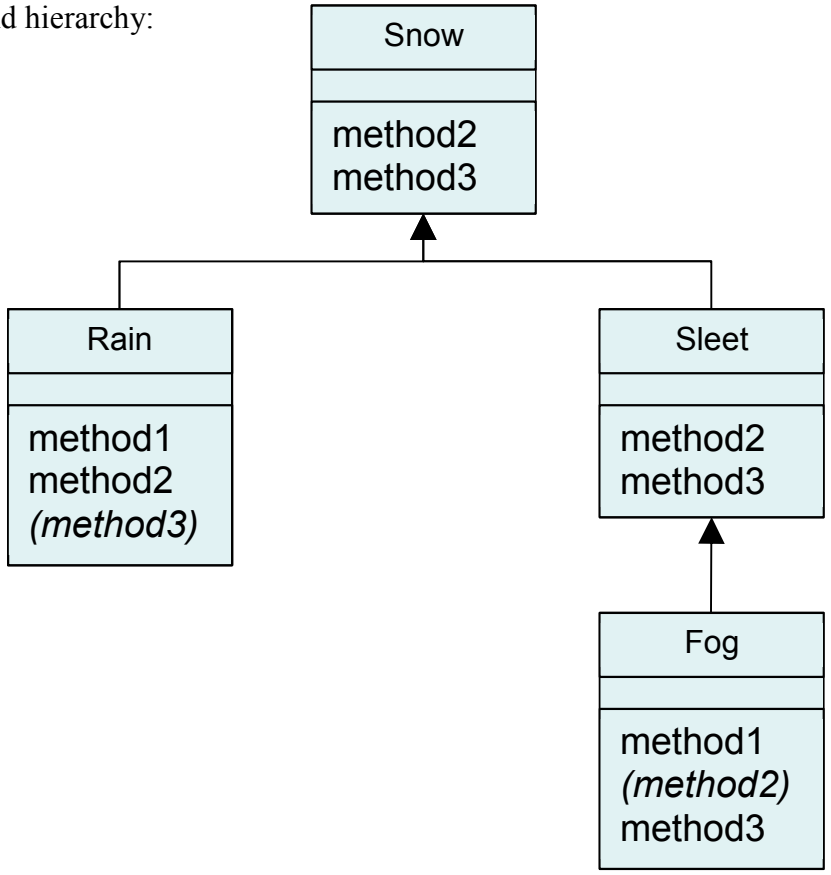
9) What is the output produced by the following recursive program? (Hint: draw stack frames)

```
public class Mystery
{
  public static void main( String[] args )
  {
    Mystery ref = new Mystery();

    System.out.println( ref.mystery( 7 ) );
  }

  public int mystery( int a )
  {
    int b = a - 3;
    int c = a + 3;

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

    return c;
  }
}
```

Output

10) Given the following class definitions and hierarchy:

```
class Snow
{
  public void method2()
  {
    System.out.println("Snow 2");
  }

  public void method3()
  {
    System.out.println("Snow 3");
  }
}

class Rain extends Snow
{
  public void method1()
  {
    System.out.println("Rain 1");
    method3();
  }

  public void method2()
  {
    super.method2();
    System.out.println("Rain 2");
    method3();
  }
}

class Sleet extends Snow
{
  public void method2()
  {
    super.method2();
    System.out.println("Sleet 2");
    method3();
  }

  public void method3()
  {
    super.method3();
    System.out.println("Sleet 3");
  }
}

class Fog extends Sleet
{
  public void method1()
  {
    System.out.println("Fog 1");
  }

  public void method3()
  {
    super.method3();
    System.out.println("Fog 3");
  }
}
```



What is the output given the following code:

```
Snow ref1;

ref1 = new Fog();

((Fog) ref1).method1();
System.out.println( "-----" );
ref1.method2();
System.out.println( "-----" );
ref1.method3();
```

Put your answer here:

What is the output given the following code:

```
Snow ref1;

ref1 = new Rain();

((Rain) ref1).method1();
System.out.println( "-----" );
ref1.method2();
System.out.println( "-----" );
ref1.method3();
```

Put your answer here:

11) Given the following partial class definition fill in the body of the constructors using the supplied comments as a guide.

```
public class Foo2 extends Foo1
{
  private Fubar var2;
  private boolean var3;

  public Foo2()
  {
                                                  // Call same class ctor passing in 420 for var1,
_____________________________________________     //   a new Fubar object invoking its no-arg ctor for
                                                  //   var2, and true for var3.
  }                                               // Assume a no-arg ctor for Fubar is defined.

  public Foo2( int var1, Fubar var2, boolean var3 )
  {

_____________________________________________     // Explicitly invoke super class (Foo1) constructor
                                                  //   passing the parameter var1. Assume ctor exists.

_____________________________________________     // Initialize the var2 instance variable by invoking
                                                  //   the copy ctor for Fubar with parameter var2.
                                                  // Assume a copy ctor for Fubar is defined.

_____________________________________________     // Initialize the boolean instance variable to the
  }                                               //   parameter var3.
}
```

Assuming class Foo1 has only one constructor, and based on the comments and your code above, write the full constructor that must be in class Foo1 and fill in the type for var.

```
public class Foo1
{
  private _____________ var;









}
```

12) Assuming class Foo1 has its one and only constructor correctly defined above, write the code the Java compiler will automatically insert in the class definition below.

```
public class Foo3 extends Foo1
{









}
```

Will this code for class Foo3 compile? Yes or No. Explain why or why not?

13) Given the following definitions:

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

```
public class Thing1 implements Doable
{
  private static final String SPEAK = "Me";

  public Thing1()
  {
    // ctor initialization here
  }

  public String speak()
  {
    return SPEAK;
  }

  public void doit()
  {
    // Thing1 does its thing
  }
}
```

```
public class Thing2 implements Doable
{
  public static final String SPEAK = "No, Me";

  public Thing2()
  {
    // ctor initialization here
  }

  public String speak( String s )
  {
    return SPEAK + s;
  }

  public void doit()
  {
    // Thing2 does its thing
  }
}
```

And the following variable definitions:

```
Thing1 thing1;
Thing2 thing2;
Doable doable;
```

Indicate which are valid Java statements. Consider each statement executed sequentially in the order it appears.

A) Valid Java statement – No Compiler Error
B) Invalid Java statement – Compiler Error

Hint: What does the compiler know about any reference variable at compile time (vs. run time)?

```
thing1 = new Thing1();      _______

thing1.speak();             _______

thing1.doit();              _______

thing1.speak( " Mine" );    _______

String s1 = Thing1.SPEAK;   _______

thing2 = new Thing2();      _______

thing2.speak();             _______

thing2.doit();              _______

thing2.speak( " Mine" );    _______
```

```
String s2 = Thing2.SPEAK;   _______

doable = new Thing1();      _______

doable.speak();             _______

doable.doit();              _______

doable = thing2;            _______

doable.speak( " Mine" );    _______

doable.doit();              _______

thing1 = thing2;            _______

thing1 = doable;            _______

doable = new Doable();      _______
```

14) Given the following class definitions:

```
abstract class Animal {
  private String name;
  public Animal() { this( "Animal" ); }
  public Animal( String name ) { this.name = name; }
  public String toString() { return this.name; }
  public abstract String speak();
}

class Cat extends Animal {
  public Cat() { this( "Brina" ); }
  public Cat( String name ) { super( name  + " Cat" ); }
  public String speak() { return "Meow"; }
  public String speak( String name ) { return name + " Meow"; }
}

class Tiger extends Cat {
  public Tiger( String name ) { super( name + " Tiger" ); }
  public String speak() { return super.speak( "Jennifer" ); }
}

class BigTiger extends Tiger {
  public BigTiger() { super( "Ko Ko" ); }
  public BigTiger( String name ) { super( name ); }
  public String speak() { return "Roar"; }
  public String speak( String name ) { return name + " Roar"; }
}

final class Lion extends Cat {
  public String speak() { return "Mo Lion " + super.speak(); }
  public String softer() { return "Marjori " + super.speak(); }
}

public class Test14 {
  public static void main( String[] args ) {

    Animal a;

    a = new Cat();
    System.out.println( a + " says " + a.speak() );

    a = new Lion();
    System.out.println( a + " says " + ((Lion) a).softer() );

    a = new BigTiger();
    System.out.println( a + " says " + a.speak() );

    a = new Tiger( "Max" );
    System.out.println( a + " says " + a.speak() );

    a = new BigTiger( "Zach" );
    System.out.println( a + " says " + ((Cat) a).speak( "Big" ) );

  }
}
```

What gets printed when this program is run?

Can we subclass/extend from Tiger like this? State Yes or No. Then explain why or why not.

```
class LittleTiger1 extends Tiger {
  public LittleTiger1() { super( "Little Tiger1" ); }
  public String speak() { return this.name + super.speak(); }
}
```

Can we subclass/extend from Animal like this? State Yes or No. Then explain why or why not.

```
class Dog extends Animal {
  public Dog() { super( "Dog" ); }
  public String speak( String name ) { return name + " says Woof"; }
}
```

Can we subclass/extend from Lion like this? State Yes or No. Then explain why or why not.

```
class CowardlyLion extends Lion {
  public String toString() { return "Courage " + super.toString(); }
}
```

Can we subclass/extend from Tiger like this? State Yes or No. Then explain why or why not.

```
class LittleTiger2 extends Tiger {
  public String speak() { return "Little " + super.speak(); }
}
```

Can we subclass/extend from Cat like this? State Yes or No. Then explain why or why not.

```
class StrayCat extends Cat {
  public String toString() { return "Stray " + super.toString(); }
}
```

Can we subclass/extend from Tiger like this? State Yes or No. Then explain why or why not.

```
class LittleTiger extends Tiger {
  public LittleTiger() { super( "Little Tiger" ); }
  public String speak( String name ) { return name + super.speak(); }
}
```

16) Consider the following program?

```
 1  public class Test16
 2  {
 3    private int a;
 4    private int b;
 5    private static int c = 7;

 6    public static void main( String[] args )
 7    {
 8      Test16 ref = new Test16( 3 );

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

11    public Test16( 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;
20      a = c;

21      System.out.println( "this.a = " + this.a );
22      System.out.println( "this.b = " + this.b );
23      System.out.println( "Test16.c = " + Test16.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( a ) );
28      System.out.println( "this.a = " + this.a );
29      System.out.println( "this.b = " + this.b );
30      System.out.println( "Test16.c = " + Test16.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.b + Test16.c;

40      x = a = b + c;

41      System.out.println( "this.a = " + this.a );
42      System.out.println( "this.b = " + this.b );
43      System.out.println( "Test16.c = " + Test16.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      Test16.c = c + 2;
49      this.a = a + c;

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

Use the numbers below to identify various program parts.

A) static method
B) class definition (type)
C) local variable
D) instance variable
E) actual argument
F) constructor
G) instance method
H) static variable
I) formal parameter

_____ `main()` on line 6
_____ `Test16` on line 1
_____ `method2()` on line 36
_____ `Test16()` on line 11
_____ `ref.b` on line 9
_____ `x` on line 40
_____ `b` on line 4
_____ `c` on line 39
_____ `c` on line 5
_____ `a` on line 11

Where in the Java Runtime environment does each of the following live?

`a` on line 3 _____
`b` on line 18 _____
`c` on line 5 _____
`x` on line 15 _____

Output

```
this.a = _________
this.b = _________
Test16.c = _________
c = _________
b = _________
a = _________
this.a = _________
this.b = _________
Test16.c = _________
x = _________
a = _________
b = _________
c = _________
result = _________
this.a = _________
this.b = _________
Test16.c = _________
x = _________
a = _________
b = _________
c = _________
```

Given the following class definitions for class Foo, class Fubar, and class FubarTest:

```
public class Foo
{
  public Foo()
  {
    System.out.println( "Foo ctor #1" );
  }

  public Foo( int x, int y )
  {
    this();
    System.out.println( "Foo ctor #2" );
  }

  public String toString()
  {
    System.out.println( "Foo.toString" );
    return "Foo";
  }
}
```

```
public class FubarTest
{
  public static void main( String[] args )
  {
    Foo ref = new Fubar();

    System.out.println( "*****" );

    System.out.println( ref.toString() );
  }
}
```

```
public class Fubar extends Foo
{
  public Fubar( int x, int y, int z )
  {
    super( x, y );
    System.out.println( "Fubar ctor #1" );
  }

  public Fubar( int x, int y )
  {
    this ( x, y, 42 );
    System.out.println( "Fubar ctor #2" );
  }

  public Fubar()
  {
    this( 4, 2 );
    System.out.println( "Fubar ctor #3" );
  }

  public String toString()
  {
    String s = super.toString() + " + " +
               "Fubar";
    System.out.println( s );
    return "Fubar.toString";
  }
}
```

17) What is the output when we run FubarTest as in
java FubarTest

Given the initial order of ints in an array as: 3, 6, 9, 8, 1, 0, 5 what is the order of the elements after 3 iterations of the selection sort algorithm? Recall the selection sort algorithm finds the index of the smallest value in the unsorted partition and exchanges (swaps) that value with the value at the index of the first element of the unsorted partition, then increments the index of the unsorted partition.

____ ____ ____ ____ ____ ____ ____

What is Rick's favorite sorting algorithm? _____

Scratch Paper

Scratch Paper