Signature _____ **Name** _____

cs11f ____ **Student ID** _____

CSE 11
Final
Fall 2011

Page 1 _____ (17 points)

Page 2 _____ (25 points)

Page 3 _____ (31 points)

Page 4 _____ (14 points)

Page 5 _____ (10 points)

Page 6 _____ (19 points)

Page 7 _____ (20 points)

Page 8 _____ (12 points)

Page 9 _____ (31 points)

Page 10 _____ (9 points)

Total _____ (188 points = 179 base points + 9 points EC [5%])
(100%)

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</u> valid Java identifiers? (Circle your answer(s).)

```
FiveStar          Five-Star         5_Star            INT

5Star             Five_Star         Five*****         integer
```

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

```
int a = 5, b = -1, c = 3;

System.out.println( !( a + b > c ) );                          ________

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

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

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

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

```
int a = 7, b = 5, 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 = 7, y = 5, 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?

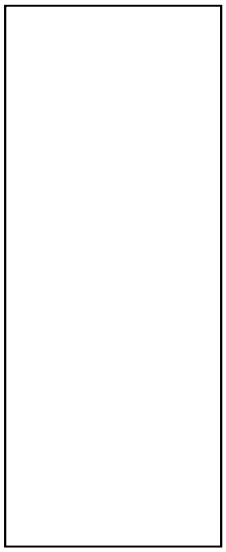
```
!( x < -10 && y != 7 )
```

_____

4) What gets printed?

```
public class Question4
{
  public static void main( String[] args )
  {
    final int MAX = 9, MIN = 3;
    int i = 6, j = 6;

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



5)

What gets printed if the value of the actual argument passed to this method is 2? _____

```
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 2? _____

```
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 );
}
```

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 = p1;
Point p3 = new Point( 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) _____________

p1.translate(1, 1);  // Add 1 to the x and y coordinates in the Point object ref'ed by p1

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

7) A bear is an animal and a zoo has many animals, including bears. Three classes Animal, Bear, and Zoo are declared to represent animal, bear, and zoo objects. Which of the following is the most appropriate set of declarations? _____

A
```
class Bear extends Animal, Zoo { ... }
```

B
```
class Animal extends Bear { ... }
class Zoo { private Animal[] zooAnimals; }
```

C
```
class Animal extends Zoo {
  private Bear zooBear;
}
```

D
```
class Bear extends Animal implements Zoo { ... }
```

E
```
class Bear extends Animal { ... }
class Zoo { private Animal[] zooAnimals; }
```

F
```
class Zoo extends Animal {
  private Bear[] zooAnimals;
}
```

8) Complete the following method which is intended to return the index of the smallest value in the array numbers. You can assume the array is not empty. If the smallest value appears more than once in the array, return the index of the last occurrence of this smallest value.

```
public static int findSmallest( int[] numbers )
{
  int smallest = _________________;

  for ( int i = _____________; ____________________________________________________; __________ )
  {
    if ( ____________________________________________________________________ )
    {
      ____________________________________________________ ;
    }
  }
  return ____________________________;
}
```

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( 6 ) );
  }

  public int mystery( int a )
  {
    int b = a - 3;
    int c = a + 5;

    if ( b > 0 )
    {
      System.out.println( a + " " + b + " " + c );
      c = b + mystery( --a );
      System.out.println( a + " " + b + " " + c );
    }
    else
    {
      c = a - b;
      System.out.println( "Whoa!" );
      System.out.println( a + " " + b + " " + c );
    }

    return c;
  }
}
```

Output

10) Given the following definition of class Thing2, what is the output of the Java application Test10?

```
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 1:
        s = s + "first ";

      case 2:
        s = s + "mid ";
        break;

      case 3:
        s = s + "last ";
        break;

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

    return s;
  }

  public void swap1( Thing2 t2 )
  {
    int temp;

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

  public void swap2( Thing2 t2 )
  {
    Thing2 temp;
    Thing2 t1 = this;

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

```
public class Test10
{
  public static void main( String[] args )
  {
    Thing2 first = new Thing2( 1 );
    Thing2 second = new Thing2( 2 );

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

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

    Thing2 third = new Thing2( 3 );
    Thing2 fourth = new Thing2( 4 );

    third.swap1( fourth );

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

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

    System.out.println( third == first );
    System.out.println( fourth == second );
    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( 1 );
    second = new Thing2( 2 );

    first.swap2( second );

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

Output

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 Fubar 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? 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( "Ko Ko" ); }
  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( "Great" ); }
}

class BigTiger extends Tiger {
  public BigTiger() { super( "Brina" ); }
  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 "Chung Lion " + super.speak(); }
  public String softer() { return "Katherine " + super.speak(); }
}

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

    Animal a;

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

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

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

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

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

  }
}
```

What gets printed when this program is run?

15) Use the class definitions on the previous page to answer the following.

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

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

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

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

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

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

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

```
class SuessLion extends Lion {
  public String toString() { return "Lion in the Hat " + super.toString(); }
}
```

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

```
class SuessCat extends Cat {
  public String toString() { return "Cat in the Hat " + super.toString(); }
}
```

16) What output is produced by the following program?

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

 6    public static void main( String[] args )
 7    {
 8      Test16 ref = new Test16( 4 );

 9      ref.method1( ref.a );
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 + 2;
20      a = c + 3;

21      System.out.println( "this.a = " + this.a );
22      System.out.println( "Test16.b = " + Test16.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( "Test16.b = " + Test16.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 + Test16.b;

40      x = a = b + c;

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

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

Use the letters below to identify various program parts.

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

_____ `method1()` on line 15
_____ `Test16()` on line 11
_____ `b + c` on line 27
_____ `main()` on line 6
_____ `Test16` on line 1
_____ `b` on line 18
_____ `a` on line 3
_____ `a` on line 11
_____ `ref` on line 8
_____ `b` on line 4

Output

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

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

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

  public Fubar()
  {
    this( 10, 20 );
    System.out.println( "Fubar ctor #2" );
  }

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

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

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

  public Foo()
  {
    this( 4, 2, 0 );
    System.out.println( "Foo ctor #3" );
  }

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

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

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

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

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

Scratch Paper

Scratch Paper