Signature _____ **Name** _____

cs11f ____ **Student ID** _____

CSE 11
Final
Fall 2010

Page 1 _____ (14 points)

Page 2 _____ (20 points)

Page 3 _____ (36 points)

Page 4 _____ (14 points)

Page 5 _____ (10 points)

Page 6 _____ (15 points)

Page 7 _____ (25 points)

Page 8 _____ (8 points)

Page 9 _____ (30 points)

Page 10 _____ (9 points)

Total _____ (181 points = 172 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).)

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

C$E11               CSE_11                   My1stJavaClass     float
```

2) Using the operator precedence table above, evaluate each expression and state what gets printed. Remember short-circuit evaluation with && and ||.

```
int i = 1, j = 2, k = 3, m = 2;

System.out.println( !( k <= m ) );                          ________

System.out.println( j <= i && j == m || k <= m );           ________

System.out.println( !(i >= 1) && j != 4 );                  ________

System.out.println( !(i > 4 || j <= 6) == i >= 4 && j > 6 ); ________
```

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

```
int a = 2, b = 6;
boolean c = !(b > 4) && (a <= 6) && (a <= 4) || (b > 6);

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

a =
b =
c =

```
int x = 2, y = 6;
boolean z = !((x > 4) || (y <= 6)) == ((y <= 4) && (x > 6));

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

x =
y =
z =

4) What gets printed?

```
public class Question4
{
   public static void main( String[] args )
   {
      final int MAX = 11;
      int i = 4, j = 8;

      for (i = 7; i <= MAX; ++i )
      {
         j = i;
         while ( j < MAX )
         {
            --j;
            System.out.println( i + " " + j );
            j += 3;
         }
      }

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

5)

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 );
}
```

6) Assume a program had the following declarations:

```
Location loc1 = new Location( 420, 42 );
Location loc2 = new Location( loc1 );
Location loc3 = loc1;
```

What results would be produced by evaluating the following expressions?

`( loc2 == loc3 )` _____ `loc2.equals( new Location( loc3 ) )` _____

`( loc1 == loc2 )` _____ `loc2.equals( loc1 )` _____

7) An interface definition is limited to having only _____ and

_____.

A concrete class cannot have _____ declared in its definition.

A _____ class cannot have any subclasses.

The keyword to denote inheritance of interface is _____.

The keyword to denote inheritance of implementation is _____.

8) Complete the following method which is intended to return the sum of all values less than the parameter `val` in the array numbers.

```
public static int sumLessThanVal( int[] numbers, int val )
{
  int sum = ____________;

  for ( int i = _____________; ____________________________________________; __________ )
  {
    if ( ____________________________________________________ )
    {
       ____________________________________________________ ;
    }
  }
  return ____________;
}
```

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

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

    System.out.println( ref.mystery( 8 ) );
  }

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

    if ( c > 0 )
    {
      System.out.println( a + " " + b + " " + c );
      c = b + mystery( --a );
      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 definition of class Thing1, what is the output of the Java application Question10?

```
class Thing1
{
  private int count;

  public Thing1( int count )
  {
    this.count = count;
  }

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

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

  public String toString()
  {
    if ( this.count == 5 )
      return "five";
    else if ( this.count == 6 )
      return "six";
    else if ( this.count == 7 )
      return "seven";
    else
      return "need more";
  }

  public void swap1( Thing1 t2 )
  {
    Thing1 temp;
    Thing1 t1 = this;

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

  public void swap2( Thing1 t2 )
  {
    int temp;

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

```
public class Question10
{
  public static void main( String[] args )
  {
    Thing1 first = new Thing1( 7 );
    Thing1 second = new Thing1( 5 );

    Thing1 temp = first;
    first = second;
    second = temp;

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

    Thing1 third = new Thing1( 5 );
    Thing1 fourth = new Thing1( 4 );;

    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 Thing1( 6 );
    second = new Thing1( 4 );

    first.swap1( 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 double var3;

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

    ______________________________________        // Explicitly invoke super class (Foo1) constructor
                                                  //   passing the parameter var1.

    ______________________________________        // Initialize the double instance variable to the
                                                  //   parameter var3.

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

  public Foo2()
  {
                                                  // Call same class ctor passing in 42 for var1,
    ______________________________________        //   a new Fubar object invoking its no-arg ctor for
                                                  //   var2, and 80.86 for var3.
  }                                               // Assume a no-arg ctor for Fubar is defined.
}
```

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.

```
public class Foo1
{
  private __________ var1;









}
```

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 abstract class MyPet
{
  public abstract String speak();
}
```

```
public class Puppy extends MyPet
{
  private static final String
                 PUPPY_SPEAK = "Bark";

  public Puppy()
  {
    // ctor initialization here
  }

  public String speak()
  {
    return PUPPY_SPEAK;
  }

  public void sleep( int time )
  {
    // puppy sleeps for time seconds
  }
}
```

```
public class Kitty extends MyPet
{
  private static final String
                 KITTY_SPEAK = "Meow";

  public Kitty()
  {
    // ctor initialization here
  }

  public String speak()
  {
    return KITTY_SPEAK;
  }

  public void wag()
  {
    // kitty wags its tail
  }
}
```

And the following variable definitions:

```
Puppy puppy;
Kitty kitty;
MyPet pet;
```

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

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

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

```
kitty = new Kitty();          _______

puppy = new Puppy();          _______

pet = kitty;                  _______

pet.speak();                  _______

pet.wag();                    _______

pet.sleep( 3000 );            _______

kitty = pet;                  _______

pet = new MyPet();            _______

pet = puppy;                  _______

pet.speak();                  _______

( (Puppy) pet).wag();         _______

( (Puppy) pet).sleep( 3000 ); _______

puppy = pet;                  _______

puppy = kitty;                _______

kitty.wag();                  _______
```

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 name; }
  public abstract String speak();
}

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

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

class BigTiger extends Tiger {
  public BigTiger() { super( "Anu Tiger" ); }
  public BigTiger( String name ) { super( name ); }
  public String speak() { return super.speak( "Anu" ); }
}

class Lion extends Cat {
  public String speak() { return "Heather Lion " + super.speak(); }
  public String softer() { return "Bruce Lion " + 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( "Ankur" );
    System.out.println( a + " says " + a.speak() );

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

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

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

  }
}
```

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 Lion like this? Explain why or why not.

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

Can we subclass/extend from Animal like this? Explain why or why not.

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

If class Lion was defined as a final class (`final class Lion extends Cat`), can we define SuessLion like this? Explain why or why not.

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

Can we make `abstract class Animal` an interface instead of a class (`interface Animal`) and change `class Cat extends Animal` to `class Cat implements Animal`? Explain why or why not.

16) What output is produced by the following program?

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

 6    public static void main( String[] args )
 7    {
 8      Test16 ref = new Test16( 5 );

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

11    public Test16( int c )
12    {
13      this.c = c;
14    }

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

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

21      System.out.println( "Test16.a = " + Test16.a );
22      System.out.println( "this.b = " + this.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( c + b ) );
28      System.out.println( "Test16.a = " + Test16.a );
29      System.out.println( "this.b = " + this.b );
30      System.out.println( "this.c = " + this.c );
31      System.out.println( "a = " + a );
32      System.out.println( "b = " + b );
33      System.out.println( "c = " + c );
34      System.out.println( "x = " + x );
35    }

36    private int method2( int x )
37    {
38      int a = x;
39      int c = this.c + Test16.a;

40      x = b = a + c;

41      System.out.println( "Test16.a = " + Test16.a );
42      System.out.println( "this.b = " + this.b );
43      System.out.println( "this.c = " + this.c );
44      System.out.println( "a = " + a );
45      System.out.println( "b = " + b );
46      System.out.println( "c = " + c );

47      Test16.a = a + 2;
48      this.b = b + c;

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

Output

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

Use the letters below to identify various program parts.

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

_____ `Test16()` on line 11
_____ `method2()` on line 36
_____ `Test16` on line 1
_____ `ref.c` on line 9
_____ `main()` on line 6
_____ `a` on line 38
_____ `c` on line 5
_____ `a` on line 3
_____ `x` on line 15
_____ `ref` on line 8

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

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

  public Foo()
  {
    System.out.println( "Foo ctor #2" );
  }

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

```
public class Fubar1 extends Foo
{
  public Fubar1( int x, int y, int z )
  {
    super( x, y );
    System.out.println( "Fubar ctor #1" );
  }

  public Fubar1( int x, int y )
  {
    this( x, y, -99 );
    System.out.println( "Fubar ctor #2" );
  }

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

```
public class FubarTest
{
  public static void main( String[] args )
  {
    Foo ref = new Fubar1( 25, 151 );

    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