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%])

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

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>! ++ --</td>
<td>right to left</td>
</tr>
<tr>
<td>* / %</td>
<td>left to right</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>left to right</td>
</tr>
<tr>
<td>== !=</td>
<td>left to right</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>right to left</td>
</tr>
</tbody>
</table>

1) Which of the following are valid Java identifiers? (Circle your answer(s).)

1stJavaClass My-First-Java-Class sEvEnTeEn CSE11Is_1_
C$E11 CSE_11 MylstJavaClass float

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

```java
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?

```java
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--;
```

```java
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--;
```
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.

```java
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)

```java
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**
Given the following definition of class Thing1, what is the output of the Java application Question10?

```java
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() );
        first.swap1( second );
        System.out.println( first.toString() );
        System.out.println( second.toString() );
    }
}
```
11) Given the following partial class definition fill in the body of the constructors using the supplied comments as a guide.

```java
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.

```java
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.

```java
public class Foo3 extends Foo1 {

}
```

Will this code for class Foo3 compile? Why or why not?
13) Given the following definitions:

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

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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitty = new Kitty();</td>
<td>B)</td>
</tr>
<tr>
<td>puppy = new Puppy();</td>
<td>B)</td>
</tr>
<tr>
<td>pet = kitty;</td>
<td>B)</td>
</tr>
<tr>
<td>pet.speak();</td>
<td>B)</td>
</tr>
<tr>
<td>pet.wag();</td>
<td>B)</td>
</tr>
<tr>
<td>pet.sleep( 3000 );</td>
<td>B)</td>
</tr>
<tr>
<td>kitty = pet;</td>
<td>B)</td>
</tr>
<tr>
<td>pet = new MyPet();</td>
<td>B)</td>
</tr>
<tr>
<td>pet = puppy;</td>
<td>B)</td>
</tr>
<tr>
<td>pet.speak();</td>
<td>B)</td>
</tr>
<tr>
<td>((Puppy) pet).wag();</td>
<td>B)</td>
</tr>
<tr>
<td>((Puppy) pet).sleep( 3000 );</td>
<td>B)</td>
</tr>
<tr>
<td>puppy = pet;</td>
<td>B)</td>
</tr>
<tr>
<td>puppy = kitty;</td>
<td>B)</td>
</tr>
<tr>
<td>kitty.wag();</td>
<td>B)</td>
</tr>
</tbody>
</table>

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

```java
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( String name ) { super( name + " Cat" ); }
    public String speak() { return "Meow"; }
}

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

class BigTiger extends 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(); }
}

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.

```java
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.

```java
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.

```java
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?

```java
public class Test16 {
    private static int a;
    private int b;
    private int c;

    public static void main( String[] args )
    {
        Test16 ref = new Test16( 5 );
        ref.method1( ref.c );
    }

    public Test16( int c )
    {
        this.c = c;
    }

    public void method1( int x )
    {
        int c = ++x;
        int b;
        b = c + 3;
        a = b + 2;

        System.out.println( "Test16.a = " + Test16.a );
        System.out.println( "this.b = " + this.b );
        System.out.println( "this.c = " + this.c );
        System.out.println( "c = ");
        System.out.println( "b = " + b );
        System.out.println( "a = " + a );
        System.out.println( "result = " + method2( c + b ) );
        System.out.println( "Test16.a = " + Test16.a );
        System.out.println( "this.b = " + this.b );
        System.out.println( "this.c = " + this.c );
        System.out.println( "a = ");
        System.out.println( "b = " + b );
        System.out.println( "c = ");
        System.out.println( "x = " + x );
    }

    private int method2( int x )
    {
        int a = x;
        int c = this.c + Test16.a;
        x = b - a + c;

        System.out.println( "Test16.a = " + Test16.a );
        System.out.println( "this.b = " + this.b );
        System.out.println( "this.c = " + this.c );
        System.out.println( "a = ");
        System.out.println( "b = " + b );
        System.out.println( "c = ");
        Test16.a = a + 2;
        this.b = b + c;
        return x + 5;
    }
}
```

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

_____ Test16() on line 11  _____ a on line 38
_____ method2() on line 36  _____ c on line 5
_____ Test16 on line 1  _____ a on line 3
_____ ref.c on line 9  _____ x on line 15
_____ main() on line 6  _____ ref on line 8
Given the following class definitions for class Foo, class Fubar, and class FubarTest:

```java
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