Midterm Examination #2
CSE 11, UCSD
Practice

RULES:

1. Don't start the exam until you are told to.

2. This is a closed-book, closed-notes, no-calculator exam. Don't refer to any materials other than the exam itself.

3. Write your name, AND your cs11 login name, on each page of the exam when you get to it.

4. Do not look at anyone else's exam. Do not talk to anyone but an exam proctor during the exam.

5. If you're wearing a billed cap, turn it so the bill is facing back, or take it off. Please turn off cell phones and pagers.

6. If you have a question, raise your hand and an exam proctor will come to you.

7. You have 1 hour and 15 minutes to finish the exam. When you are done, give your exam to a proctor. The proctor will check your picture ID and sign the ID check below.

8. Your exam grade will be emailed to your cs11 account within 2 working days. Exams will be handed back in special session this week.

#	Topic	max pts	actual pts
1.	General T/F	24	
2.	General F/B	24	
3.	Loops	12	
4.	Inheritance	28	
5.	Ctors	12	
	TOTAL	100	

grader:_____

PROCTOR ID CHECK:_____

1. [24 pts; 1 pt each correct answer; -1 pt each incorrect answer] **T**rue or **F**alse:

   a. __F____ In Java boolean expressions, the Object pointer value `null` is equivalent to `false`.

   b. __T____ When called, a public instance method of an object can access private instance variables of that object.

   c. __T____ If not overridden, public methods of a base class become public methods of a derived class.

   d. __T____ In Java, the type `Boolean` is a subclass of `Object.`

   e. __T____ If you want a method to return a value, you must put a "`return`" statement somewhere in it.

   f. __F____ Constructors are inherited.

   g. __T____ A constructor of a class must never be declared to return `void`.

   h. __T____ It is impossible for a Java class to have more than one parent class.

   i. __F____ To perform an iteration until a complicated boolean condition becomes false, it is usually better to use a `for` statement than a `while` statement.

   j. __T____ Overloading a method means defining a method to have the same name but a different number or type of arguments.

   k. __F____ If you do not first use the setLayout() method to specify a layout manager, no components can be added to a Swing container.

   l. __F____ A constructor method for a class is called when a variable of that class type is declared.

   m. __T____ A variable declared in the "init" part of a for-loop is visible within that for-loop only.

   n. __T____ "Instance of a class" and "object of a class" mean the same thing.

   o. __T____ The BorderLayout layout manager for a container will resize components if the container is resized.

   p. __T____ In Java, a class may implement any number of interfaces.

   q. __F____ If you attempt to use a DataInputStream object to access a text file, an IOException will be thrown.

   r. __T____ Java uses dynamic binding to determine which version of an overridden method will be called.

   s. __T___ In a JFC component, x coordinates increase going to the right, and y coordinates increase going downward.

   t. __T____ As a programmer, to understand how to use a class, you should read the documentation of what the public methods of the class do.

   u. __T____ Java primitive type values are automatically copied when passed as arguments to methods.

   v. __T____ A feature of the exception handling mechanisms in Java is that they let the programmer separate the tasks of handling normal cases and handling exceptional cases.

   w. __F____ Literal constants of type String can be used as case labels in a Java switch statement.

   x. __T____ Every Java while-loop can be re-written as a for-loop that does exactly the same thing.

2. [21 pts, 3pts ea blank] Fill in the blank with the best word or words that completes the sentence.
   a. ____Inheritance (or derivation)___ is the acquisition by one class of the variables and methods of another class.
   b. The Java class that is not a subclass of any other class is _____Object_____.
   c. A(n) ____JPanel_____ is the best JFC component class to use for doing general graphics.
   d. If a method definition is declared___static (or public static)_____, it can be called using just the class name, without creating an object.
   e. A constructor with no arguments is called a(n) ____default_____ constructor.
   f. If a class is declared ____final_____ in its definition, the class cannot be used as a base class.
   g. The style of programming in which user interactions with GUI objects determine the flow of execution in your program is called _____event-driven_____ programming.
   h. A class defined within the body of another class definition is called a(n)_____inner (or nested)_____ class.

3. [12 pts, 3pts ea blank] For each of the following, write down the number of lines printed, and the largest number printed. (There are no infinite loops or syntax errors in these examples.)
   a.

```
for(int i=0; i<10; i++) {
   for(int j=0; j<100; j++) {
      if(j==41) continue;
      System.out.println(j);
   }
}
```

   Number of lines printed:_____990_____ Largest number printed: _____99_____

   b.

```
int n=10;
try {
   while (n < 50) {
      if(n == 27) throw new Exception();
      System.out.println(n);
      n = n + 3;
   }
} catch (Exception e) {
   System.out.println(n);
}
```

   Number of lines printed:___14_____ Largest number printed: _____49_____

4. [28 pts; 4 pts each] Consider these Java class definitions:

```
-----------------------------------------------------
public class X {
      private int a;
      private double b;
      public X () { a = 1; b = 2; }
      public X ( int num1, int num2 ) { a = num1; b = num2; }
      public double getVal () { return b; }
      public double getOtherVal (X x) { return x.getVal(); }
}
-----------------------------------------------------
public class Y extends X {
      public int c;
      public Y ( int num ) { c = num; }
      public double getVal () { return c; }
      public int amount (X x) { return (int)(getOtherVal(x) + 7); }
}
-----------------------------------------------------
```

Now consider the marked statements in the Java code below. For each one, write down what would be printed when the statement executes; or write "ERROR' if the statement would cause a compile-time or runtime error.

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

      X x_obj1 = new X(5,6);
      X x_obj2 = new X(-4,-5);
      Y x_obj3 = new Y(2);
```

a. __2_____ `System.out.println( x_obj3.c );`

b. __6.0_____ `System.out.println( x_obj1.getVal() );`

c. __2.0_____ `System.out.println( x_obj3.getVal() );`

d. __-5.0_____ `System.out.println( x_obj3.getOtherVal(x_obj2) );`

e. __2.0_____ `System.out.println( x_obj2.getOtherVal(x_obj3) );`

f. __ERR_____ `System.out.println( x_obj2.amount(x_obj3) );`

g. ___2____ `System.out.println( x_obj3.amount(x_obj2) );`

5. [12 pts; 3pts each correct answer, -3pts each incorrect answer] Consider the following code:

```
-----------------------------------------------------
public class A {
    private int x;
    public A () { ??? }
    public A (int x) { ??? }
}
-----------------------------------------------------
public class B extends A {
    public B () { ??? }
    public B (int x) { ??? }
}
-----------------------------------------------------------
public class Program {
public static void main ( String args[] ) {
    A a_obj = new A(4);
    a_obj = new A();
    B b_obj = new B();
    b_obj = new B(6);
    }
}
```

When Program's main() method is executed, the following four numbers (and nothing else) are printed out on separate lines: first 4, then 6, then 4, and finally 6.
For each of the following definitions for the constructors, write "YES" if they work as required, else "NO".

a._ YES____
```
public A () {}
public A (int x) {System.out.println(x);}
public B() {super(6);}
public B(int x) {super(x-2); System.out.println(x);}
```

b.__NO____
```
public A () { System.out.println(x);}
public A (int x) {this.x = x+2; System.out.println(x); }
public B () {super(4); }
public B (int x) { System.out.println(x); }
```

c.._NO___
```
public A () {System.out.println(6);}
public A (int x) {System.out.println(x);}
public B() {System.out.println(4);}
public B(int x) {System.out.println(x);}
```

d.___NO___
```
public A () {System.out.println(x+2);}
public A (int x) {this.x = x; }
public B() {System.out.println(4);}
public B(int x) {super(x);}
```