CIS 121 - Exam 1 Solution - Wednesday, June 23, 2004

Name: ________________________________ J Number: ________________________________

I. Fill in the Blank (30 pts)

1. If a class definition doesn’t contain an extends clause, then it extends **Object**.

2. If a local variable has the same name as an instance variable, then the local variable is said to **shadow** the instance variable.

3. When all references to an object are lost, we call that object **garbage**.

4. The least restrictive access control keyword is **public**. (That is, a variable defined with this keyword is accessible anywhere.)

5. **Encapsulation** is the binding together of data and the code which operates on the data.

6. **Method Overriding** involves changing the implementation of a method that’s inherited.

7. An object refers to its immediate superclass with the keyword **super**.

8. The field **length** is associated with an array and determines how many elements are in the array.

9. The process of **double buffering** involves drawing a complete image in memory before it is displayed in the visible area.

10. In order for an applet to run in a Thread, it can implement the Runnable interface which contains the method **public void run()**.

11. The **signature** of a method is a combination of its name and parameter list.

12. When class A extends B, we call B a **superclass** of A.

13. The operator `==` between two object references indicates whether the references are **aliases** which means that they point to the same object in memory.
14. An object can use the keyword **this** to refer to itself.

15. If a class definition is declared **final**, then we can’t create subclasses of that class.

16. The operator **instanceof** between an object reference and a class definition indicates whether the object reference refers to an object which was created from the class definition or one of its superclasses.

17. The interpreter uses **short-circuit evaluation** to evaluate expressions with multiple logical conditions which means that if the truth or falsehood of the expression can be determined by evaluating one of the expressions beginning at the left, then it will not evaluate the remaining expressions.

18. The method found in the ActionListener interface is **public void actionPerformed(ActionEvent e)**.

19. The parameters listed in the declaration of a method are called its **formal** parameters.

20. (True/False) **False** JButton is a subclass of Button.

21. If a class is declared **abstract**, then we can’t create instances of it with a constructor.

22. A **String literal** is created by surrounding 0 or more characters within double quotes.

23. We use the keyword **static** to distinguish a class variable from an instance variable.

24. **Information Hiding** is the ability to restrict the access to or the visibility of data.

25. We call the method **getContentPane()** within an applet to give us a reference to that part of the applet where components can be placed.

26. **CLASSPATH** is an environment variable containing a list of subdirectories to search for class definitions when using the import statement.

27. Strings are **immutable** which means that once they are created, they can’t be changed.

28. Non-primitive data types are sent to methods by **reference** instead of by value.

29. A **constructor** is used in a class definition to initialize its instance variables, and this method has the same name as the class.

30. Java supports **single inheritance** which means that we can’t have more than one class definition in an extends clause.
II. Matching (10 pts) Please choose the most appropriate choice for each term from the candidates on the right.

31. g.drawOval(0,0,getWidth(),getHeight()) ☛ A. Draws a circle whose radius is 1/4 the width of an applet.

in an applet where g is a Graphics object

32. ActionListener ☛ B. The type of object expected by the setPreferredSize method.

33. Sequence,Selection, and Repetion ☛ C. A relationship between A and B that allows objects of type A
to be treated as if they are objects of type B.

34. Tag Interface ☛ D. An example of method overloading and not method overriding.

35. public String toString(String s) ☛ E. Draws a circle centered at the center of the applet.
in a class definition

36. A “is a” ☛ F. A level of access control that allows instance variables only to be
accessed in the class in which they are defined.

B. The type of object expected by the setPreferredSize method.

37. protected ☛ G. A relationship between A and B that allows objects of type B
to be treated as if they are objects of type A.


39. s.charAt(s.length()) ☛ I. A singing trio from the 1960s that frequently appeared on the Sonny

where s is a reference to a String

and Cher show.

40. ClassCastException ☛ J. An example of a compile-time error and not a run-time error.

K. An interface that contains no variables or methods but acts as an

indication to the compiler and interpreter that the class definition

has a certain capability.

L. Contains the method public void itemStateChanged(ItemEvent e).

M. A level of access control that allows instance variables to be accessed
in the same class, the same package, and in any subclass.

N. An object containing the x and y coordinates of a point.

O. An example of method overriding and not method overloading.

P. An example of a run-time error not involving casting instead of a

compile-time error.

Q. The three types of control structures that Bohm and Jacopini

showed we can write all programs using.

R. Can occur when we try to cast an object reference to a subclass
reference when the object reference is not compatible with that subclass.

S. A level of access control that allows instance variables to only be
accessed in the same class and in the same package.

T. Can occur when we try to cast a subclass reference to a superclass
reference.

U. A level of access control that allows instance variables to be accessed
anywhere.

V. An interface which is tagged unusable.

W. An example of a compile-time error and not a compile-time error.
III. Short Answer (30 pts)

41. (5 pts) Suppose we have an array named *words* in which each entry is the String literal “Hello”. What is printed by the following code? Explain.

```java
System.out.println(words[0] == words[words.length-1]);
```

**Because only one copy of a String literal is maintained in memory, true is printed**

42. (5 pts) Suppose we have a class definition called MyClass. Assume it contains a constructor that accepts one int. Suppose it’s immediate superclass is a class named MySuperclass which contains a constructor that accepts no arguments. Which of the following assignment statements is correct syntax? Explain.

a) MyClass myclass = new MySuperclass();

b) MySuperclass mySuperclass = new MyClass(1);

**Recall that all MyClass objects are MySuperclass objects, but not all MySuperclass objects are MyClass objects. So b is correct**

43. (5 pts) Suppose s is a String reference. We want to create a second String referred to by s1 which will contain all of the characters of the original String referred to by s except for the first character and the last character. Show a line of code that uses one method call to do this.

```java
s1 = s.substring(1, s.length()-1);
```

44. (5 pts) Within an applet’s paint method, we want to draw a String that is centered horizontally and is drawn at vertical position 50. Given a reference, *metrics*, to a FontMetrics object associated with the Graphics object in the paint method, how would you draw this String?

```java
g.drawString(string, getWidth()/2-metrics.stringWidth(string)/2, 50)
```
45. (5 pts) Suppose we have the int variable a assigned to the number of primitive data types in Java, the int b assigned to the number of class definitions that can appear in an extends clause, the int c assigned to the number of characters in the class name of the immediate superclass of JApplet, and the int d assigned to the value of new int[5].length*2. What is the value of (((a+b)*c) 

\[
\begin{align*}
\text{a} &= 8 \\
\text{b} &= 1 \\
\text{c} &= 6 \\
\text{d} &= 10 \\
\text{(((a+b)*c \% (c-b)) + a/b) &= (((8+1)*6 \% 5) + 8 = (9*6 \% 5) + 8 = 54 \% 5 + 8 = 4 + 8 = 12}
\end{align*}
\]

46. (2 pts) Suppose we create 1,375,763 instances of a class definition which contains a class variable called number. How many copies of number are maintained in memory?

1

47. (2 pts) What are two conditions that must be satisfied to allow us to override a method we inherit?

- We can’t change the return type
- The method can’t be declared final in the superclass
- We can’t make the access control keyword more restrictive

48. (1 pt) Considering what you know about abstract classes and final classes, would it make any sense to try to create a class definition that was both final and abstract? Explain.

The only reason we make a class abstract is so that we can make subclasses of it. But if the class is final, we can’t make subclasses of it. So it wouldn’t make sense to try to make a class both abstract and final
IV. Problem Solving and Coding (30 pts)

49. (6 pts) Consider the following two class definitions. Is there a syntax error when we try to compile Subclass.java? If not, what is printed when we interpret Subclass?

```java
public class Question49 {
    public int number;
    public Question49(int number) {
        this.number = number;
    }
    public Question49() {
        number = 1;
    }
}
```

```java
public class Subclass extends Question49 {
    private int number1;
    public Subclass(int number1) {
        this.number1 = number1;
    }
    public String toString() {
        return(number1 + "");
    }
    public static void main(String[] args) {
        Subclass subclass = new Subclass(2);
        System.out.println(subclass);
    }
}
```

Since we don't explicitly call a superclass constructor from Question49, a call is made to its no-argument constructor. This sets the value of number to 1. That is added to the number sent to the subclass constructor. So what is printed is 3

50. (6 pts) Will the following class definition compile correctly? If not, what is the problem? If it does compile correctly, what happens when it is interpreted?

```java
public class Question50 {
    private int number;
    public Question50(int number) {
        this.number = number;
    }
    public int number() {
        return(number);
    }
    public boolean equals(Object o) {
        return(number == o.number());
    }
    public void static main(String[] args) {
        Question50 question50 = new Question50(1);
        Question50 question51 = new Question50(1);
        System.out.println(question50 == question51);
    }
}
```

This class definition will not compile correctly because in the equals method, the parameter is promoted to be an Object. So we can't call the method number on the parameter because it is not defined in Object.
51. (6 pts) What occurs when the following applet is loaded? Is anything printed on the applet’s window? If not, explain the problem. Read the code carefully.

```java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Question51 extends JApplet implements ActionListener {
    private JButton button;
    private String message;
    public void init() {
        Container container = getContentPane();
        container.setLayout(null);
        container.setBackground(Color.white);
    }
    public void paint(Graphics g) {
        super.paint(g);
        String[] tokens = message.split(“ ”);
        for (int counter=0; counter<tokens.length; counter++)
            g.drawString(tokens[counter], 20, 20 + counter*20);
    }
    public void actionPerformed(ActionEvent e) {
        message = “This is a test”;
        repaint();
    }
}
```

Recall that the sequence of method calls when an applet loads is init, start, and paint. Since message isn’t initialized in init and since there isn’t a constructor, message is null. So there will be a run-time exception when the paint method is called.

For questions 52 and 53, consider the following class definition.

```java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public abstract class Superclass extends JApplet implements ActionListener, ItemListener {
    protected Container container;
    public void init() {
        container = getContentPane();
        container.setLayout(null);
        container.setBackground(Color.white);
    }
    // This method will return a String with all of
    // the vowels, a, e, i, o, and u from the parameter
    // removed.
    public abstract String removeVowels(String s);
    // In this method, you are to fill in the lower triangular
    // region of a rectangle. The triangular region has
    // corners at the upper-right hand corner of the rectangle,
    // the lower-left hand corner of the rectangle, and the
    // lower-right hand corner of the rectangle.
    public abstract void fillTriangularRegion(Graphics g,
        int x,
        int y,
        int width,
        int height);
}
```
52. (6 pts) Show the complete implementation of a subclass of Superclass which will display the output of the removeVowels method on the String “Hello”. (You may not simply place the output on the applet. The output must be returned from removeVowels.) Make sure you include the necessary code in the subclass so that it compiles.

```java
import java.awt.*;
import java.awt.event.*;

public class Question52 extends Superclass {
    public String removeVowels(String s) {
        String output = "";
        for (int counter=0; counter<s.length(); counter++)
            if (s.charAt(counter) != 'a' &&
                s.charAt(counter) != 'A' &&
                s.charAt(counter) != 'e' &&
                s.charAt(counter) != 'E' &&
                s.charAt(counter) != 'i' &&
                s.charAt(counter) != 'I' &&
                s.charAt(counter) != 'o' &&
                s.charAt(counter) != 'O' &&
                s.charAt(counter) != 'u' &&
                s.charAt(counter) != 'U')
                output += s.charAt(counter);
        return output;
    }
    public void paint(Graphics g) {
        g.clearRect(0, 0,getWidth(),getHeight);
        super.paint(g);
        g.drawString(removeVowels("Hello"),50,50);
    }
    public void fillTriangularRegion(Graphics g,
        int x,
        int y,
        int width,
        int height) {
    }
    public void actionPerformed(ActionEvent e) {
    }
    public void itemStateChanged(ItemEvent e) {
    }
}
```
53. (6 pts) Show the complete implementation of a subclass of Superclass which will fill in the triangular portion of the applet’s window where the corners of the triangle are the upper-right corner, lower-left corner, and lower-right corner of the applet’s window. You must do this with the fillTriangular region method. Recall that methods we have used to draw in an applet. Make sure you include the necessary code in the subclass so that it compiles.

```java
import java.awt.*;
import java.awt.event.*;

public class Question53 extends Superclass {
    public String removeVowels(String s) {
        return("");
    }

    public void fillTriangularRegion(Graphics g,
            int x,
            int y,
            int width,
            int height) {
        for (int counter=x;counter<width+x;counter++)
            g.drawLine(x+width,0,counter,height);
    }

    public void paint(Graphics g) {
        g.clearRect(0,0,getWidth(),getHeight());
        super.paint(g);
        fillTriangularRegion(g,0,0,getWidth(),getHeight());
    }

    public void actionPerformed(ActionEvent e) {
    }

    public void itemStateChanged(ItemEvent e) {
    }

}