

Q1: Which of the following is not a benefit of using Swing GUI components?

- Most Swing components are heavyweight.
- Most Swing components are written completely in Java
- Swing components allow the user to specify a uniform look and feel across all platforms.
- Swing components allow the user to change the look and feel while the program is running.

Q 2: What is leJOS?

- a graphical front end for LEGO MINDSTORMS™
- an operating system and Java-based language for the RCX
- an IDE for the LEGO MINDSTORMS™ programming language
- a toolkit for building LEGO MINDSTORMS™ robots

Q3: A well-designed group of constructors:

- Guarantee the object is created in a consistent state.
- May call common methods.
- Allows the class's user flexibility in specifying some or all of the initial instance variable values.
- All of the above.

Q4. InputStream and OutputStream are \_\_\_\_\_.

- The top-level abstract classes defined in the java.io package, from which all other stream classes inherit.
- The top-level abstract classes defined in the java.io package, from which all other byte stream classes inherit.
- Classes that you can instantiate to read and write bytes.
- Interfaces that define methods that can be used to read and write bytes.

Q5. You are writing a class Employee as a subclass of Person, i.e: public class Employee extends Person You want to call the default constructor of the Person class. Which statement should you use?

- super();
- Person();
- this();
- you can not call it.

Q6: To catch an exception, code must be enclosed in

- throws block.
- catch block.
- try block.
- finally block.

Q7: **StringBuffer** objects can be used in place of **String** objects if:

- The string data is not constant.
- The string data size may grow.
- Performance is not critical.
- All of the above.

Q8: The term encapsulation refers to:

- public** methods.
- Hiding implementation details from clients of a class.
- Accessing static class members.
- The process of releasing an object for garbage collection.

Q 9. Given a Graphics object g, to draw an outline of a rectangle of width 20 and height 50 with the upper-left corner at (20, 20), you use \_\_\_\_\_.

- g.drawRect(20, 50, 20, 20)
- g.drawRectFill(20, 20, 20, 50)
- g.drawRect(20, 20, 20, 50)
- g.drawRectFill(20, 50, 20, 20)

Q10: Which of the following refers to a "has a" relationship instead of an "is a" relationship?

- A student to person.
- A university to an establishment.
- A university to a student.
- A professor to a faculty member.

Q11: For which of the following would polymorphism not provide a clean solution?

- A billing program where there is a variety of clients who are billed with different fee structures.
- A maintenance log program where a variety of machine data is collected and maintenance schedules are produced for each machine based on the data collected.
- A program to compute savings account interest.
- An IRS program that maintains information on a variety of taxpayers and determines who to audit based on criteria for classes of taxpayers.

Q 12: Consider the Java segment:

```
String line1 = new String( "c = 1 + 2 + 3" ) ;
StringTokenizer tok = new StringTokenizer(
line1 );
```

```
int count = tok.countTokens();
```

The value of count is:

- 8.
- 7.
- 13.
- 4.

Q 13. The \_\_\_\_\_ method is executed when the page becomes inactive.

- init()
- start()
- stop()
- destroy()

Q 14. Java applications and applets both \_\_\_\_\_.

- have a main() method
- are executed using the java command
- are compiled using the javac command
- are executed from the HTML file

Q15: How many **String** objects are instantiated by the following code segment (not including the literals)?

```
String s1, output;
s1 = "hello";

output += "\nThe string
reversed is: " ;

for ( int i = s1.length() -
1; i >= 0; i-- )
    output += s1.charAt( i ) +
" " ;
a. 2.
b. 1.
c. 4.
d. 5.
```

Q16: For a method to be invoked polymorphically, the subclass:

- Must implement the method with the same signature as the superclass.
- Must inherit the method from the superclass.
- Can implement the method with any signature.
- Can implement the method with the same signature or can inherit the method from the superclass.

Q17: Which of the following is not a specific GUI component (control or widget)?

- String.
- Label.
- Menu.
- List.

Q 18: The layout manager that allows alignment to be controlled is:

- FlowLayout.**
- BorderLayout.**
- GridLayout.**
- None of the above.

Q 19: **FlowLayout** is:

- An **abstract** class.
- A way of organizing components vertically.
- The most basic layout manager.
- Left-aligned by default.

Q 20: To draw on an applet, the programmer must access the \_\_\_\_\_ object in the applet's **paint** method.

- drawString**
- drawLine**
- Graphics**
- Pixel**

Q 21: What is the entry point for a leJOS program running on the RCX?

- public static void main(String[] args)
- public static void initRCXProgram()
- public static void mainRCX(RCXParams[] args)

Q 22: The code in a finally block:

- Is always executed if the corresponding try block is entered.
- Is executed only if an exception occurs.
- Is executed only if an exception does not occur.
- Is executed only if there are no catch blocks.

Q 23: Which of the following statements about try blocks is true?

- The try block must be followed by at least one catch block.
- The try block must be followed by a finally block.
- The try block should contain statements that may throw an exception.
- The try block should contain statements to process an exception.

Q 24. Which of the following is not an advantage of Java exception handling?

- Java separates exception handling from normal processing tasks.
- Exception handling improves performance.
- Exception handling makes it possible for the caller's caller to handle the exception.
- Exception handling simplifies programming because the error-reporting and error-handling code can may be placed at the catch block.

Q 25 Which of the following statements is correct to create a BufferedWriter stream to write to a file named out.dat?

- BufferedWriter bw = new BufferedWriter(new File("out.dat"));
- BufferedWriter bw = new BufferedWriter(new FileWriter("out.dat"));
- BufferedWriter bw = new BufferedWriter("out.dat");
- BufferedWriter bw = new BufferedWriter(new FileOutputStream("out.dat"));

Put your answers Here:

1.	2.	3.	4.	5.
6.	7.	8.	9.	10.
11.	12.	13.	14.	15.
16.	17.	18.	19.	20.
21.	22.	23.	24.	25.

[Concepts and comprehension, 12 points] Answer the following questions.

C. 1 What happens if you try to compile code that looks like this: How do you fix it?

```
final class A { ...}  
class B extends A { ... }
```

C.2 Shape is an abstract class defined as follows:

```
abstract class Shape {  
    abstract void draw();  
    abstract void move(int x, int y);  
    abstract void scale(double factor);  
    int x,y;  
    int width, height;  
}
```

What happens if you try to compile the following class definition?

```
class Square extends Shape {  
    void move(int x, int y){this.x = x; this.y = y;};  
}
```

C.3 Switch is an interface defined as follows:

```
interface Steerable {  
    public void turnLeft();  
    public void turnRight();  
    public void driveForward();  
    public void applyBrakes();  
}
```

What happens if you try to compile the following class that implements the interface?

```
import josx.platform.rcx.*;  
Public class MyCar extends Thread implements Steerable {  
    public void turnLeft( ){  
        Motor.B.backward();  
        Thread.sleep(200);  
        Motor.B.stop();  
    }  
}
```

C.4 What is wrong with the following class definition?

```
class C {  
    int x = 0;  
    public static void main(String[] args){  
        System.out.println(x);  
    }  
}
```

**Tracing:[10]**

**T 1.1** What is the output from the following program?

```

class Test {
    public static void main(String[] args){
        B b = new B();
        A a = b;
        System.out.println(a.m()+b.m());
    }
}
class A {
    public String m(){
        return "A";
    }
}
class B extends A {
    public String m(){
        return "B";
    }
}

```

**T 1.2**

<pre> class Point {     public int x;     public int y;     public Point (int x, int y)     {         this.x = x;         this.y = y;     }      public void print()     {         System.out.println(""+x+","+y+"");     } } class Size {     public int w;     public int h;     public Size (int w, int h)     {         this.w = w;         this.h = h;     }      public void print()     {         System.out.println(""+w+","+h+"");     } } </pre>	<pre> class Rectangle {     Point point;     Size size;     public Rectangle (Point p, Size s)     {         point =p;         size = s;     }      public void move (Point p)     {         point = p;     }      public void resize (Size s)     {         size = s;     }      public void print()     {         point.print();         size.print();     }      public static void main (String [] args)     {         Point p1 = new Point (1,1);         Point p2 = new Point (2,2);         Size s1 = new Size (10,10);         Size s2 = new Size (20,20);         Rectangle r = new Rectangle (p1, s1);         r.print();         r.move(p2);         r.resize(s2);         p2.x =3;         p2.y = 3;         r.print();     } } </pre>
--	--

**Programming:**

1.[GUI , 9 points] The following questions and Figure refer to the program shown below that converts temperatures

between Celsius and Fahrenheit. When the user types the enter key at a text field, the program converts the value in that text field to the other scale and shows it in its text field.



Fill box A, B, C and D with appropriate statement/statements that attach event listeners to the text fields. [Hints: Pay attention to the comment above the box

```
// Temperature.java: Convert Celsius to Fahrenheit and vice versa
// Temperature.java: Convert Celsius to Fahrenheit and vice versa
import java.awt.*;
import java.awt.event.*;
public class Temperature extends Frame implements ActionListener
{
    private TextField jtfCelsius = new TextField(10);
    private TextField jtfFahrenheit = new TextField(10);
```

```
// Main method
public static void main(String[] args)
{
    Temperature frame = new Temperature();
    frame.pack();
    frame.setTitle("Temperature");
```

```
//Make Frame visible
```

```
A 
```

```
}
public Temperature()
```

```
{
    // Panel p to hold labels and text fields
    Panel p = new Panel();
```

```
//set Grid Layout 2 rows and 2 colomns
```

```
//Add a new label called Celsius
```

```
B
```

```
//Add jtfCelsius textfield object to the panel
```

```
//Add a new label called Fahrenheit
```

```
//Add jtfFahrenheit textfield object to the panel
```

```
// Add panel object to the frame
```

```
add(p);
```

```
// Register listener
```

```
C
```

```
// Handle ActionEvent
```

```
public void actionPerformed(ActionEvent e)
```

```
{  
D  
  
}
```

```
void convertc2f(){  
    String c = jtfCelsius.getText();  
    double f = Double.parseDouble(c)*1.8+32;  
    jtfFahrenheit.setText(Double.toString(f));  
}  
void convertf2c(){  
    String f = jtfFahrenheit.getText();  
    double c = (Double.parseDouble(f)-32)/1.8;  
    jtfCelsius.setText(Double.toString(c));  
}
```

```
}
```

2. [9] Suppose that a text file score.txt contains an unspecified number of scores. Write a program that reads the scores from the file, displays the score and displays the average of the scores. Scores are separated by blanks. (Hint: Read the scores one line at a time until all lines are read. For each line, use StringTokenizer to extract the scores, and convert them into double values using the Double.parseDouble method)