Name:

1.124 Quiz 2

Thursday November 9, 2000

Time: 1 hour 20 minutes
Answer all questions. All questions carry equal marks.

Question 1.
Show the steps that are involved in sorting the string *SORTME* using the quicksort algorithm given below.

```
#include <iostream.h>
void quicksort(char *a, int l, int r);

main() {
   char str[8] = "9SORTME";        // 9 is a sentinel.
   quicksort(str, 1, 6);
}

inline void swap(char *a, int i, int j) {
   char tmp = a[i];
   a[i] = a[j];
   a[j] = tmp;
   cout << a+1 << endl;                  // Print out the array, excluding the sentinel.
}

void quicksort(char *a, int l, int r) {
   if (r > l) {
      char v = a[r];
      int i = l - 1;
      int j = r;
      while (1) {
         while(a[++i] < v);
         while(a[--j] > v);
         if (j <= i)
            break;
         swap(a, i, j);
      }
      swap(a, r, i);

      quicksort(a, l, i-1);
      quicksort(a, i+1, r);
   }
}
```

Answer:

S	O	R	T	M	E
E	O	R	T	M	S
E	O	R	M	T	S
E	O	R	M	S	T
E	M	R	O	S	T
E	M	O	R	S	T

Question 2.
Show how you would translate the bold portions of the following C++ code into Java.

```
#include <iostream.h>

class Shape {
private:
   float x, y;

public:
   Shape(float a, float b) {
      x = a;
      y = b;
   }

   virtual float compute_area() = 0;

   virtual void print() {
      cout << x << " " << y << endl;
   }
};

class Circle : public Shape {
private:
   float radius;

public:
   Circle(float a, float b, float r) : Shape(a, b) {
      radius = r;
   }

   float compute_area() {
      return 3.14f * radius * radius;
   }

   void print() {
      cout << radius << endl;
      Shape::print();
   }
};


void main() {
   Circle a(3,4,2);
   a.print();
}
```

Answer:

(a)
```
public abstract float compute_area();
```

(b)
```
class Circle extends Shape
```

(c)
```
public Circle(float a, float b, float r) {
        super(a, b);
        radius = r;
    }
```

(d)
```
public void print() {
        System.out.println(radius);
        super.print();
    }
```

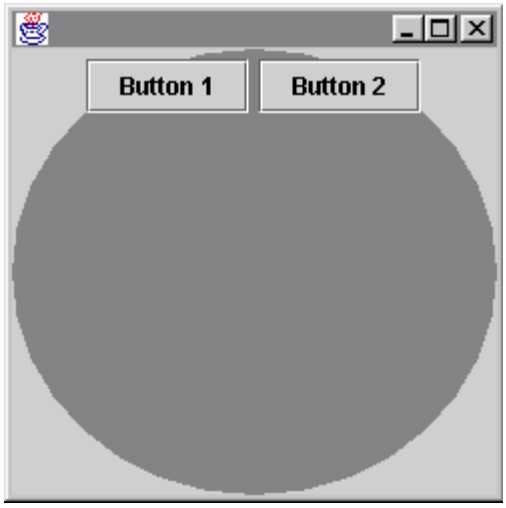
Question 3.
In the following C++ program, the *outputData()* function can handle callbacks such as *plot()* and *print()*. How would you complete the given Java code to implement a similar capability?

```
#include <iostream.h>

class Point {
private:
   int x, y;

public:
   Point(int a = 0, int b = 0) {
      x = a;
      y = b;
   }

   void print() {
      cout << x << " " << y << endl;
   }
};

typedef void (*OutFunc)(Point& p);

void plot(Point &p) {                    // Assume that this plots the point p on the screen.
   cout << "In plot:" << endl;
   p.print();
}

void print(Point &p) {                   // Assume that this prints out the coordinates of p.
   cout << "In print:" << endl;
   p.print();
}

void outputData(OutFunc pFunc, Point *a, int n) {
   for (int i = 0; i < n; i++)
      pFunc(a[i]);
}

void main() {
   Point a[2];
   a[0] = Point(2,3);
   a[1] = Point(4,5);
   outputData(plot, a, 2);
   outputData(print, a, 2);
}
```

Answer:

```
class Point {
   private int x, y;

   public Point(int a, int b) {
      x = a;
      y = b;
   }

   void print() {
     System.out.println(x + " " + y);
   }
}

interface Output {
   void out(Point p);
}

class Plotter implements Output {
   public void out(Point p) {
      // Assume that this plots the point p on the screen.
      System.out.println("In plot:");
      p.print();
   }
}

class Printer implements Output {
   public void out (Point p) {
      // Assume that this prints out the coordinates of p.
      System.out.println("In print:");
      p.print();
   }
}

class Main {
   static void outputData(Output o, Point[] a, int n) {
      for (int i = 0; i < n; i++)
         o.out(a[i]);
   }

   public static void main(String args[]) {
      Point a[] = new Point[2];
      a[0] = new Point(2,3);
      a[1] = new Point(4,5);

      outputData(new Plotter(), a, 2);
      outputData(new Printer(), a, 2);
   }
}
```

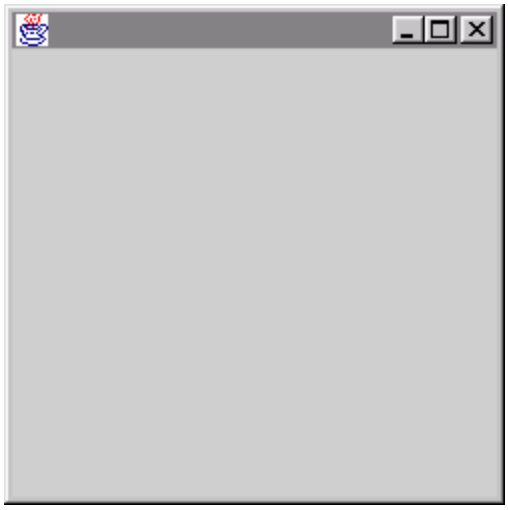
Question 4.
Show how you would complete the given Java code, so that it achieves the effect shown in the Figure below.



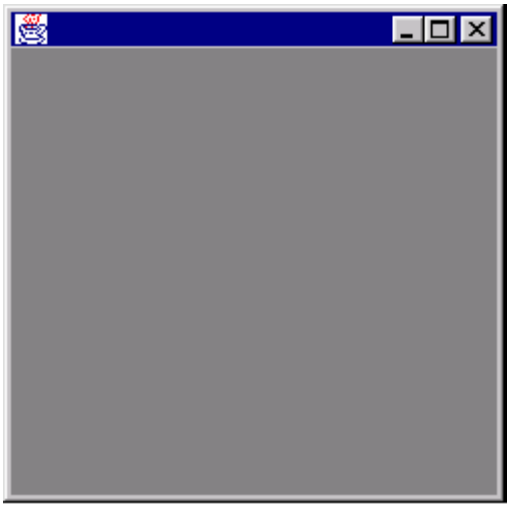
Answer:

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

class Main {
 public static void main(String args[]) {
   JFrame f = new JFrame();

   JPanel p = new JPanel() {
     public void paintComponent(Graphics g) {
          super.paintComponent(g);
          Dimension d = getSize();
          g.setColor(Color.gray);
          g.fillOval(0,0,d.width,d.height);
     }
   };

   p.setLayout(new FlowLayout());
   p.add(new JButton("Button 1"));
   p.add(new JButton("Button 2"));

   f.getContentPane().setLayout(new BorderLayout());
   f.getContentPane().add(p, BorderLayout.CENTER);

   f.setSize(250,250);
   f.setVisible(true);
 }
}
```

Question 5.
How you would you change the background color of the panel when the mouse moves over the application's window?



Mouse out



Mouse over

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

class Main {
 public static void main(String args[]) {
   JFrame f = new JFrame();
   final JPanel p = new JPanel();
```

Answer:

```
  final Color c = p.getBackground();
  p.addMouseListener(new MouseAdapter() {
    public void mouseEntered(MouseEvent e) {
        p.setBackground(Color.gray);
    }
    public void mouseExited(MouseEvent e) {
        p.setBackground(c);
    }
  });
```

```
   f.setContentPane(p);
   f.setSize(250,250);
   f.setVisible(true);
 }
}
```

Question 6.
The following applet contains several errors. Explain what changes you would make to correct the code, so that the applet displays the current frame number.

Answer:

```
/*
  <APPLET CODE=MyApplet.class WIDTH=250 HEIGHT=100>
  </APPLET>
*/

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

public class MyApplet extends JApplet implements Runnable
{
   Thread t = null;
   int count = 0;

   public void init() {
      getContentPane().add(new JPanel() {
            public void paintComponent(Graphics g) {
               super.paintComponent(g);
               g.drawString("Count = " + count, 50, 50);
            }
         });
   }

   public void start() {
      t = new Thread(this);
      t.start();
   }

   public void run() {
      for (int i = 0; i < 200; i++) {
         count++;
         repaint();
         try {
            Thread.sleep(100);
         }
         catch (InterruptedException e) {}
      }
   }
}
```

Question 7.
What is double buffering and why is it important in animation? How do the Swing classes differ from the AWT classes in this respect?

Answer:

Double buffering is important because it helps to produce a smooth looking animation that does not flicker. Without double buffering, noticeable flicker will usually result due to the clearing of the background before an animation frame is drawn. Double buffering is implemented by performing all drawing operations for an animation frame (including clearing the background) in an off-screen image buffer. Once the complete animation frame has been assembled off-screen, the contents of the image buffer are copied on to the screen.

The Swing package provides automatic double buffering capabilities by way of the *JComponent* class. With the AWT classes, double buffering is the responsibility of the programmer.

Question 8.
Show how you would extract the number *1.124* from the string *"Hello 1.124 World!"* and then store it in a *float* variable.

Answer:

```
import java.util.*;

class Main {
   public static void main(String args[]) {
      String str = "Hello 1.124 World!";

      StringTokenizer t = new StringTokenizer(str);
      String s = t.nextToken();
      s = t.nextToken();
      float f = Float.parseFloat(s);
   }
}
```