

1.00/1.001

Introduction to Computers and Engineering Problem Solving

Quiz 2 / November 10, 2005

Name:	
Email Address:	
TA:	
Section:	

You have 90 minutes to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary files have already been imported.
Good luck.

<i>Question</i>	<i>Points</i>
Question 1	<i>/ 24</i>
Question 2	<i>/ 30</i>
Question 3	<i>/ 20</i>
Question 4	<i>/ 16</i>
Question 5	<i>/ 10</i>
Total	<i>/ 100</i>

HW1	HW2	HW3	HW4
HW5	HW6	Q1	

Question 1 True/False

- 1) A class may extend one or more classes.

TRUE FALSE

- 2) A class may implement one or more interfaces.

TRUE FALSE.

The following code fragment is associated with problems 3) and 4)

```
public abstract class SomeClass{
    public SomeClass(){
        //implementation not shown
    }
    //remainder of implementation not shown
}

public interface SomeInterface{
    //declarations not shown
}
```

- 3) The statement
`SomeClass c = new SomeClass();`
will not compile in Java.

TRUE FALSE

- 4) The statement
`SomeInterface i = new SomeInterface();`
will not compile in Java.

TRUE FALSE

- 5) Events in Java are objects that are created by Event Sources (such as JButtons, JPanels, and other Components).

TRUE FALSE

- 6) Each Event Listener (for example, a class that implements the ActionListener interface) can only register to receive Events from one Event Source (for example, one JButton).

TRUE FALSE

7) Multiple different Event Listeners can register to receive Events from the same Event Source.

TRUE FALSE

8) You want to extend JPanel to make a new class that draws a smiley face in the middle of the JPanel. Where should you put your drawing code? Circle all that choices that apply:

- a) the repaint() method
- b) the constructor
- c) the paintComponent(Graphics g) method
- d) none of the above

Question 2 - Inheritance and Interfaces

QuizClass, QuizClass2 and QuizInterface are all in the same package. However, QuizClass and QuizInterface are incomplete. Read and complete the code fragments below for QuizClass and QuizInterface.

//QuizClass.java

```
public abstract class QuizClass{
    protected double var1;
    public static final int CONS1=10;
    private float var2;

    public QuizClass(double var1, float var2){
        /*Part 1: complete the constructor,
        initializing the var1 and var2 data members*/
        

this.var1 = var1;
            this.var2 = var2;


    }

    public double getVar1(){
        return var1;
    }

    public float getVar2(){
        return var2;
    }
}
```

```
/*Part 2: declare an abstract calc1() method that
has no arguments and returns a double*/
```

```
public abstract double calc1();
```

```
}
//remainder of implementation is not shown
```

```
//QuizInterface.java
```

```
public interface QuizInterface{
    double CONS2 = 3.75;
```

```
/*Part 3: complete the interface with the signature
of a single method called calc2() that has no
arguments and returns a float*/
```

```
public float calc2();
```

```
}
```

Now, read the QuizClass2 code and answer the questions that follow.

```
//QuizClass2.java
```

```
public class QuizClass2 extends QuizClass implements
    QuizInterface {

    private double var3;

    public QuizClass2(double var1, float var2, double
        var3){
        this.var1=var1;
        this.var2=var2;
        this.var3=var3;
    }
    //remainder of the implementation is not shown
}
```

- 4 As written, the QuizClass2 constructor will cause a compiler error. Describe the cause of the error below and correct it within the code fragment on the next page.

There are 2 errors; either one gained full points. First, since there is no default constructor for the super class, super (var1, var2); must be called. Once that is done, this.var2=var2; would cause an error since var2 is private in QuizClass and cannot be accessed directly.

//QuizClass2.java - to be corrected below

```
public class QuizClass2 extends QuizClass implements
    QuizInterface {

    private double var3;

    public QuizClass2(double var1, float var2, double
        var3){

        super(var1,var2);
        this.var3 = var3;

    }
    //remainder of the implementation is not shown
}
```

- 5 Name two methods that must be implemented in QuizClass2.

Method 1: public double calc1()

Method 2: public float calc2()

- 6 List all the data members of `QuizClass2` that are either declared within it or inherited from one or more of its base classes.

[var1, var3, CONS1, CONS2 \(, var2\). Whether var2 is really a data member of](#)

[QuizClass2 is a bit of a philosophical question.](#)

Question 3 - Writing an ActionListener (18 points)

Owattagoo Siam wants to write a class that extends JPanel, and has one JButton that exits whatever program the JPanel is used in. He decides to call the class ExitPanel. He knows how to add the JButton, but doesn't understand Swing Event Handling.

Below is Owattagoo's ExitPanel class, but it's missing all the code to make the exit JButton functional. Fill in the code that he's missing. Note that there may be extra empty space below— you don't have to fill in all of it. There are at least two correct ways to do this, and they use code in different places. Choose the way that is easiest for you. (Hint: use ActionListener)

```
public class ExitPanel extends JPanel implements ActionListener
{

    private JButton exitButton;

    // constructor
    public ExitPanel() {
        exitButton = new JButton("Exit");
        this.setLayoutManager(new BorderLayout(this,
            BorderLayout.X_AXIS));
        this.add(exitButton);


exitButton.addActionListener(this);

// alternative solution (inner class)
exitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e){
        System.exit(0);
    }
});

    }
}
```

NOTE: There are 2 solution to this problem. One is in red, the other is in blue. Many people lost points for mixing between the two solutions, which will not work (unless you fully implement both).

```
// This method contains code to exit the program. Call
// this method, or just write this
// line of code elsewhere if that's easier.
public void exitProgram() {
    System.exit(0);
}
```

```
// implement this method for the ActionListener interface
// if the first box was filled in.
public void actionPerformed(ActionEvent e) {

    // some people checked the source of e – great, but not
    // required.

    // exit the program
    exitProgram();
}
```

```
}
```

Question 4 - Numerical methods & matrices

A symmetric matrix is a square matrix where the entries are symmetric with respect to the main diagonal (top left to bottom right entries).

The following matrix is an example of a symmetric matrix.

$$\begin{bmatrix} 1 & 3 & 5 \\ 3 & 7 & 9 \\ 5 & 9 & 11 \end{bmatrix} \rightarrow \text{Main diagonal}$$

You should now write the code to construct a symmetric matrix from a given square matrix. The code should be **general**, i.e. it should work for an $n \times n$ square matrix, and it should be efficient, in the sense that it should avoid any unnecessary computations. You should convert the square matrix into a symmetric matrix by changing the elements in the upper triangle (elements above the main diagonal), based on the lower triangle

For example, if you are given the square matrix $\begin{bmatrix} 1 & 2 & 4 \\ 3 & 7 & 6 \\ 5 & 9 & 11 \end{bmatrix}$,

the symmetric matrix you should construct would be $\begin{bmatrix} 1 & 3 & 5 \\ 3 & 7 & 9 \\ 5 & 9 & 11 \end{bmatrix}$

Fill in the blanks in the `makeSymmetricMatrix()` method with the code that will allow you to do this.

```
public class SquareMatrix {  
  
    private int n;  
    int[][] a;  
  
    public SquareMatrix()  
    {  
        a = new int[n][n];  
    }  
}
```

```
public class SymmetricMatrix {
```

```
    public static SymmetricMatrix makeSymmetricMatrix (SquareMatrix sm)  
    {
```

```
        //Your code goes here  
There are 2 ways you can do this (we gave full credit to both ways, even though  
the first is more efficient than the second)  
1. //Changing inner loop limits  
        for (int i=0;i<sm.a.length-1;i++)  
            for (int j=i+1;j<sm.a.length;j++)  
            {  
                sm.a[i][j] = sm.a[j][i];  
            }  
  
2. //Testing for upper or lower half (i<j or i>j)  
        for (int i=0;i<sm.a.length; i++)  
            for (int j=0;j<sm.a.length;j++)  
            {  
                if (i>j)  
                    sm.a[j][i]= sm.a[i][j];  
            }  
Note: In case you tested for (i<j), you must set  
a[i][j]=a[j][i]
```

```
    }  
}
```

Question 5 - Stacks

You are given the following implementation of the GenericArrayStack from class:

```
public class GenericArrayStack<E> implements GenericStack<E>
{
    static public final int DEFAULT_CAPACITY = 8;
    private E[] stack;
    private int top = -1;
    private int capacity;

    public GenericArrayStack(int cap)
    {
        capacity = cap;
        stack = (E[]) new Object[capacity];
    }

    public GenericArrayStack()
    {
        this( DEFAULT_CAPACITY );
    }

    public boolean isEmpty()
    {
        return top < 0;
    }

    public void push(E e)
    {
        if (++top == capacity)
            grow();
        stack[top] = e;
    }

    public E pop()
    {
        if ( isEmpty() )
            throw new EmptyStackException();
        else
        {
            E e = stack[top];
            stack[top--] = null;
            return e;
        }
    }
}
```

The following class uses GenericArrayStack to perform a mathematical operation specified in main(). Read the class code and then specify the program output in the empty box at the end.

```
public class StackFactor {  
  
    public static void main(String[] args) {  
        GenericStack<Integer> fS = factor( 88 );  
        while ( !fS.isEmpty() )  
        {  
            int f = fS.pop();  
            System.out.println( f );  
        }  
    }  
  
    public static GenericStack<Integer> factor( int i ) {  
        GenericStack<Integer> stack =  
            new GenericArrayStack<Integer>();  
        int j = 2;  
        while ( i > 1 )  
        {  
            if ( i % j == 0 )  
            {  
                i = i / j;  
                stack.push( j );  
            }  
            else  
                j++;  
        }  
        return stack;  
    }  
}
```

11
2
2
2