

**Electronic, Electrical and Computer Engineering**

**EE2E1 Class Test 16th Jan 2014, 2pm**

**Instructions**

This test contains 20 multiple choice questions (1-20) and 4 other questions (21-24).

The maximum number of marks obtainable is 30.

Examination conditions apply.

The time allowed for the test is **45 minutes**.

Questions 1-20 are 1 mark each.

Questions 21-24 are for 10 marks in total.

Marks are not deducted for incorrect answers.

Any calculator which does NOT store textual information may be used.

You may NOT use a mobile phone, personal organiser, or personal audio device for any purpose, including as a calculator.

All answers must be written in the spaces provided on the question paper. Pen or pencil may be used.

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mark /30 : \_\_\_\_\_\_\_\_\_\_**

1. Java I/O is based on the concept of \_\_\_\_\_\_\_

Classes

****Streams

Buffers

Threads

2. \_\_\_\_\_\_\_ is the keyword used to inherit a class

inherits

****implements

extends

super

3. An abstract class can be declared as final

True

False

4. Java supports multiple inheritance

True

False

5. Methods declared as static have several restrictions: 1) They can only call other static methods. 2) They must only access static data. 3) They cannot refer to *this*. Which of these are TRUE?

1 and 2

1 and 3

2 and 3

1, 2 and 3

6. The data, or variables, defined within a class are called \_\_\_\_\_\_\_ variables

Object

Class

Instance

None of the above

7. Java allows objects to initialize themselves when they are created using \_\_\_\_\_\_\_\_\_

Arguments

Classes

Constructors

Parameters

8. The *new* operator dynamically allocates ­­­­­­­­­­­**\_\_\_\_\_\_\_\_\_\_**  for an object and returns a reference to it.

Classes

Variables

Memory

None of the Above

9. Method overloading is one of the ways that Java supports \_\_\_\_\_\_\_\_\_\_\_\_

Encapsulation

Classes

Inheritance

Polymorphism

10. An interface contains **\_\_\_\_\_\_\_\_\_\_** methods

Abstract

Non-abstract

Implemented

Un-implemented

11. You read the following statement in a Java program that compiles and executes:

submarine.dive(depth);

Which one of these can you say for sure?

depth must be an int

dive must be a method.

dive must be the name of an instance field.

submarine must be the name of a class

submarine must be a method.

12. Which of the following layout manager is the default layout manager for the *Frame* class in Java.

FlowLayout

BorderLayout

GridLayout

CardLayout

GridBagLayout

13. What does the following code draw? g.setColor(Color.black); g.drawLine(10, 10, 10, 50); g.setColor(Color.RED); g.drawRect(100, 100, 150, 150);

A red vertical line that is 40 pixels long and a red square with sides of 150 pixels

A black vertical line that is 40 pixels long and a red square with sides of 150 pixels

A black vertical line that is 50 pixels long and a red square with sides of 150 pixels

A red vertical line that is 50 pixels long and a red square with sides of 150 pixels

A black vertical line that is 40 pixels long and a red square with sides of 100 pixel

14. State true or false :- Constructors can be overloaded like regular methods.

True

False

15. What will be the result of compiling the following code: public class Test { static int age; public static void main (String args []) { age = age + 1; System.out.println("The age is " + age); } }

Compiles and runs with no output

Compiles and runs printing out The age is 1

Compiles but generates a runtime error

Does not compile

Compiles but generates a compile time error

16. What is the result of expression 5.45 + "3.2"?

The double value 8.6

The string ""8.6"

The long value 8.

The String "5.453.2

17. When Java source code is compiled, each individual class is put into its own output file named after the class and using the \_\_\_\_\_\_ extension

.h

. java

>java

.class

18. \_\_\_\_\_\_\_\_\_\_\_\_ is a highly optimized set of instructions designed to be executed by the Java run-time system

Byte Code

Firewall

Tetra Code

View Code

19. What a Java programmer calls a \_\_\_\_\_\_\_\_\_\_ , a C/C++ programmer calls a function

Classes

Method

Function

Object

20. For the following piece of Java code, public class MyClass{public MyClass(){/\*code\*/}// more code...}, to instantiate MyClass, you would write?

MyClass mc = new MyClass();

MyClass mc = MyClass();

MyClass mc = MyClass;

MyClass mc = new MyClass;

The next 2 questions refer to the following complete piece of Java code.

public class EmployeeTest

{

public static void main(String[] args)

{

Employee[] staff ;

staff[0] = new Employee("John Brown", 35000, new Date(1989,10,1));

staff[1] = new Employee("Tony Smith", 38000, new Date(1990,3,15));

int i;

for (i = 0; i < 2; i++) staff[i].raiseSalary(5);

for (i = 0; i < 2; i++) staff[i].print();

}

}

class Employee

{

public Employee(String n, double s, Date d)

{

name = n; salary = s; hireDay = d;

}

public void print()

{ System.out.println(name + " " + salary ); }

public void raiseSalary(int byPercent)

{ double salary \*= (1 + byPercent / 100); }

private String name;

private double salary;

private Date hireDay;

}

21. There are at least 4 programming errors with this code. State what they are. **Marks will only be awarded for the first 4 errors identified.**

[4 marks]

**1. Memory for array items in array *staff*  not allocated**

**2. Don’t need to declare *salary* as double in *raiseSalary***

**3. Integer divide (byPercent/100) will yield zero**

**4. Need to import *java.util* for the *Date* class**

22. *Date(…)*  is a built in type. Write down the function header for the *Date* constructor.

[1 mark]

**public Date(int y, int m, int d);**

The next 2 questions refer to the following piece of Java code. This piece of code is a GUI which toggles the background colour of the button panel between yellow and red by clicking on the button. A screenshot of the GUI is also shown.

class ButtonPanel extends JPanel implements ActionListener

{

public ButtonPanel()

{

toggleButton = new JButton("Toggle background colour");

add(toggleButton);

setBackground(Color.yellow);

// **A.** Insert missing statement here

}

public void actionPerformed(ActionEvent evt)

{

// **B.** Insert code here

}

private JButton toggleButton;

}

class ButtonFrame extends JFrame

{

public ButtonFrame()

{

setTitle("ButtonTest");

setSize(300, 200);

addWindowListener(new WindowAdapter()

{ public void windowClosing(WindowEvent e)

{ System.exit(0); }

} );

Container contentPane = getContentPane();

contentPane.add(new ButtonPanel());

}

}

public class ButtonTest

{

public static void main(String[] args)

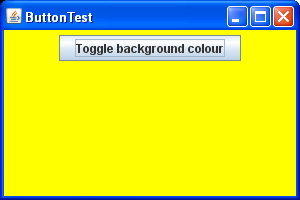
{

JFrame frame = new ButtonFrame();

frame.setVisible(true);

}

}



23. Add the missing statement indicated at comment A in the code.

[1 mark]

**toggleButton.addActionListener(this);**

24. Add the code for the *actionPerformed* method as at comment B in the code.

[4 marks]

**if (evt.getSource()==toggleButton)**

**{**

**if (getBackground()==Color.yellow)**

**setBackground(Color.red);**

**else if (getBackground()==Color.red)**

**setBackground(Color.yellow);**

**}**