Hall Ticket Number:

II/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

February, 2023 **Common to CS/CB/DS/IT Branches**

Third Semester **Object Oriented Programming**

Time: Three Hours **Maximum:**70 Marks

Answer Question No.1 compulsorily. (14X1 = 14 Marks)
Answer ONE question from each unit. (4X14=56 Marks)

1.	a)	What is byte code file in Java?	CO1	L1	1M
	b)	Define final keyword in Java.	CO1	L1	1M
	c)	Why 'main ()' method is declared as ***public static*** in JAVA programming?	CO1	L1	1M
	d)	Why multiple inheritance is not possible in Java with classes?	CO2	L1	1M
	e)	Differentiate StringBuffer and StringBuilder classes.	CO2	L4	1M
	f)	What is an Abstract class?	CO2	L1	1M
	g)	Define user defined Exception.	CO3	L1	1M
	h)	Draw the thread life cycle diagram.	CO3	L1	1M
	i)	What is the purpose of PrintWriter class?	CO3	L1	1M
	j)	Define Thread Synchronization.	CO3	L1	1M
	k)	Define Applet.	CO4	L1	1M
	l)	List any four AWT components.	CO4	L1	1M
	m)	What are the different Mouse Events?	CO4	L2	1M
	n)	What is JTree?	CO4	L1	1M
		Unit – I			
2.	a)	Define Polymorphism and Explain how to implement it with an example program.	CO1	L1	7M
	b)	When to use a Static variable in JAVA programming? Explain the importance of Static variable with a JAVA program.	CO1	L3	7M
		(OR)			
3.	a)	List and explain JAVA buzzwords. Which factors are making JAVA famous Language?	CO1	L2	7M
	b)	Explain the usage of **"this"** keyword in JAVA.	CO1	L2	7M
		Unit - II			
4.	a)	Explain Abstract classes and methods with suitable example.	CO2	L2	7M
	b)	Demonstrate the usage of Access Modifiers in JAVA Packages.	CO2	L1	7M
		(OR)			
5.	a)	Write a JAVA program to demonstrate String handling methods.	CO2	L3	7M
	b)	List and Explain any two Collection classes in JAVA.	CO2	L1	7M
		Unit - III			
6.	a)	Explain the difference between creating a thread by extending **Thread** class and implementing **Runnable** interface with an example program.	CO3	L1	7M
	b)	Write a Java program to demonstrate User defined Exceptions.	CO3	L2	7M
		(OR)			
7.	a)	Write a JAVA program to copy the contents of one file to another file.	CO3	L4	7M
	b)	Explain multi-threading. Write the purpose of isAlive() and join() functions in JAVA. Explain the same with an example.	CO3	L2	7M
		Unit - IV			
8.	a)	Discuss various AWT containers with examples.	CO4	L3	7M
	b)	Write a short note on the following i) JTextArea ii) JTable	CO4	L2	7M
		(OR)			
9.	a)	Explain key events with an example program.	CO4	L2	7M
	b)	Write a program to demonstrate any Layout Manager.	CO4	L3	7M

1. **a) What is byte code file in Java?** **1M**

Bytecode file contains an intermediate code generated by the compiler after the compilation of source code

b) Define final keyword in Java. **1M**

The final keyword used to declare constants and to prevent inheritance and method overriding.

c) Why 'main ()' method is declared as *public static* in JAVA programming? **1M**

- The main method is public in Java because it has to be invoked by the JVM. So, if main() is not public in Java, the JVM won't call it.
- The main() method is declared static so that JVM can call it without creating an instance of the class containing the main() method.

d) Why multiple inheritance is not possible in Java with classes? **1M**

In java, multiple inheritance is not supported because of ambiguity problem.

e) Differentiate StringBuffer and StringBuilder classes. **1M**

StringBuffer	StringBuilder
Thread safe	Not Thread safe
Synchronized	Non-Synchronized
Slower	Faster

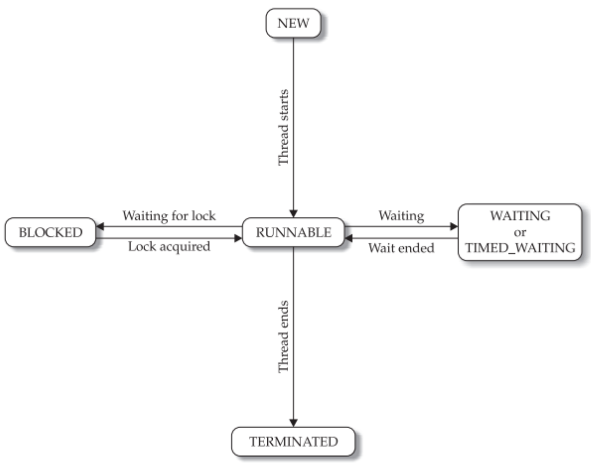
f) What is an Abstract class? **1M**

Abstract classes are classes that contain one or more abstracted behaviors or methods.

g) Define user defined Exception. **1M**

Java user-defined exception is a custom exception created and throws that exception using a keyword 'throw'. It is done by extending a class 'Exception'.

h) Draw the thread life cycle diagram. **1M**



i) **What is the purpose of PrintWriter class?** **1M**

Prints formatted representations of objects to a text-output stream

j) **Define Thread Synchronization.** **1M**

Synchronization in Java is the capability to control the access of multiple threads to any shared resource.

k) **Define Applet.** **1M**

Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document.

l) **List any four AWT components.** **1M**

TextField

Label

Button

Panel

Note: Give full marks for any 4 correct AWT components.

m) **What are the different Mouse Events?** **1M**

MOUSE_CLICKED

MOUSE_ENTERED

MOUSE_EXITED

MOUSE_PRESSED

MOUSE_RELEASED

MOUSE_MOVED

MOUSE_DRAGGED

n) **What is JTree?** **1M**

The JTree class is used to display the tree structured data or hierarchical data.

UNIT – I

2. a) Define Polymorphism and Explain how to implement it with an example program. **7M**

Polymorphism means method having more than one form. (or) The same method can perform different operations in different scenarios.
In Java polymorphism can be achieved in two ways:

1. Method Overriding
2. Method Overloading

Method Overloading:

```
class Adder{
	static int add(int a,int b){
		return a+b;
	}
	static int add(int a,int b,int c){
		return a+b+c;
	}
}

class Overloading{
	public static void main(String[] args){
		System.out.println(Adder.add(11,11));
		System.out.println(Adder.add(11,11,11));
	}
}
```

Note: Give full marks for any other correct example program.

Method Overriding:

```
class Parent {
	void show()
	{
		System.out.println("Parent's show()");
	}
}
class Child extends Parent {
	void show()
	{
		System.out.println("Child's show()");
	}
}
class Main {
	public static void main(String[] args)
	{
		Parent obj1 = new Parent();
		obj1.show();
		Parent obj2 = new Child();
		obj2.show();
	}
}
```

Note: Give full marks for any other correct example program.

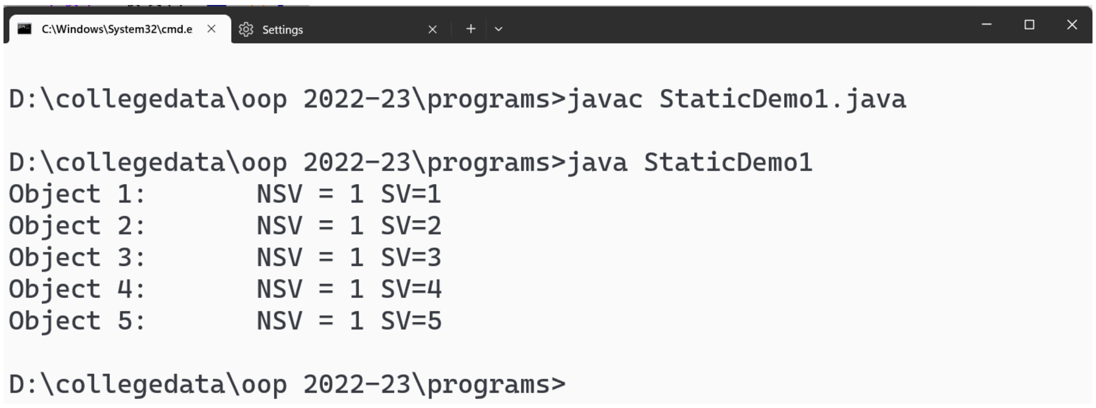
b) When to use a Static variable in JAVA programming? Explain the importance of Static variable with a JAVA program. **7M**

The static variable can be used to refer to the common property of all objects (which is not unique for each object). The advantage of static variable is the static variable gets memory only once in the class area at the time of class loading and all the objects of that class will use same memory location.

<u>Example Program:</u>

```
class StaticDemo1 {
        int nsv = 0;
        static int sv = 0;
        public static void main(String[] args){
                StaticDemo1 sd[] = new StaticDemo1[5];
                for(int i=0;i<5;i++){
                        sd[i] = new StaticDemo1();
                        sd[i].nsv++;
                        sd[i].sv++;
                        System.out.print("Object "+(i+1)+":\t");
                        System.out.println("NSV = "+sd[i].nsv+"\tSV="+sd[i].sv);
                }
        }
}
```

<u>Output:</u>



```
D:\collegedata\oop 2022-23\programs>javac StaticDemo1.java

D:\collegedata\oop 2022-23\programs>java StaticDemo1
Object 1:       NSV = 1 SV=1
Object 2:       NSV = 1 SV=2
Object 3:       NSV = 1 SV=3
Object 4:       NSV = 1 SV=4
Object 5:       NSV = 1 SV=5

D:\collegedata\oop 2022-23\programs>
```

Note: Give full marks for any other correct example program.

(OR)

3. a) List and explain JAVA buzzwords. Which factors are making JAVA famous Language? **7M**

- Simple
- Secure
- Portable
- Object-oriented

- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Simple

Java was designed to be easy for the professional programmer to learn and use effectively. Assuming that you have some programming experience, you will not find Java hard to master. If you already understand the basic concepts of object-oriented programming, learning Java will be even easier. Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java.

Object-Oriented

Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean, usable, pragmatic approach to objects. Borrowing liberally from many seminal object-software environments of the last few decades, Java manages to strike a balance between the purist's "everything is an object" paradigm and the pragmatist's "stay out of my way" model. The object model in Java is simple and easy to extend, while primitive types, such as integers, are kept as high-performance nonobjects.

Robust

The multiplatformed environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas to force you to find your mistakes early in program development. At the same time, Java frees you from having to worry about many of the most common causes of programming errors. Because Java is a strictly typed language, it checks your code at compile time. However, it also checks your code at run time. Many hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java. To better understand how Java is robust, consider two of the main reasons for program failure: memory management mistakes and mishandled exceptional conditions (that is, run-time errors). Memory management can be a difficult, tedious task in traditional programming environments.

Multithreaded

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things simultaneously. The Java run-time system comes with an elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems. Java's easy-to-use approach to multithreading allows you to think about the specific behavior of your program, not the multitasking subsystem.

Architecture-Neutral

A central issue for the Java designers was that of code longevity and portability. At the time of Java's creation, one of the main problems facing programmers was that no guarantee existed that if you wrote a program today, it would run tomorrow—even on the same machine. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever." To a great extent, this goal was accomplished.

Interpreted and High Performance

As described earlier, Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be executed on any system that implements the Java Virtual Machine. Most previous attempts at cross-platform solutions have done so at the expense of performance. As explained earlier, the Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler. Java run-time systems that provide this feature lose none of the benefits of the platform-independent code.

Distributed

Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. Java also supports Remote Method Invocation (RMI). This feature enables a program to invoke methods across a network.

Dynamic

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustness of the Java environment, in which small fragments of bytecode may be dynamically updated on a running system.

Security and Portability are the two significant factors that make java a popular language.

b) Explain the usage of "this" keyword in JAVA. 7M

The “**this”** keyword refers to the current object in a method or constructor.
The most common use of the “**this”** keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).
this can also be used to:

- Invoke current class constructor
- Invoke current class method
- Return the current class object
- Pass an argument in the method call
- Pass an argument in the constructor call

Example Program:

```
public class Main {
        int x;
        public Main(int x) {
                this.x = x;
        }
        public static void main(String[] args) {
                Main myObj = new Main(5);
                System.out.println("Value of x = " + myObj.x);
        }
}
```

Note: Give full marks for any other correct example program.

UNIT – II

4. a) Explain Abstract classes and methods with suitable example. 7M

Abstract classes are classes that contain one or more abstracted behaviors or methods.
Abstract method can only be used in an abstract class, and it does not have a body.

Example Program:

```
abstract class Multiply {
        public abstract int MultiplyTwo (int n1, int n2);
        public abstract int MultiplyThree (int n1, int n2, int n3);
        public void show() {
                System.out.println ("Method of abstract class Multiply");
        }
}
```

```
class AbstractMethodEx1 extends Multiply {
        public int MultiplyTwo (int num1, int num2) {
                return num1 * num2;
        }
        public int MultiplyThree (int num1, int num2, int num3) {
                return num1 * num2 * num3;
        }
        public static void main (String args[]) {
                Multiply obj = new AbstractMethodEx1();
                System.out.println ("Multiplication of 2 numbers: " +
                                                obj.MultiplyTwo (10, 50));
                System.out.println ("Multiplication of 3 numbers: " +
                                                obj.MultiplyThree (5, 8, 10));
                obj.show();
        }
}
```

Note: Give full marks for any other correct example program.

b) Demonstrate the usage of Access Modifiers in JAVA Packages. **7M**

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

The summary of access modifiers is shown in below table.

	Private	No Modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

(OR)

5. a) Write a JAVA program to demonstrate String handling methods. 7M

Program:

```
class StringDemo{
        public String toString(){
                return "String class methods";
        }
        public static void main(String[] args){
                String s1 = "Bapatla";
                String s2 = "Bapatla";
                String s3 = new String(s1);

                System.out.println("s1="+s1+"\ts1.length="+s1.length());
                String s4 = "Hello, "+s1;
                System.out.println("s4="+s4+"\ts4.length="+s4.length());

                System.out.println("s1.compareTo(s4)="+s1.compareTo(s4));
                System.out.println("s4.regionMatches(7,s1,0,7)="+
                                                        s4.regionMatches(7,s1,0,7));
                System.out.println("s1.charAt(2):"+s1.charAt(2));
                char[] chars = new char[s4.length()];
                s4.getChars(0,s4.length(),chars,0);
                for(char ch:chars){
                        System.out.print(ch+"\t");
                }
                System.out.println();
                StringDemo sd = new StringDemo();
                System.out.println("StringDemo.toString()"+sd.toString());
        }
}
```

Note: Give full marks for any 10 methods.

b) List and explain any two Collection classes in JAVA. 7M

Any group of individual objects which are represented as a single unit is known as the collection of the objects. In Java, a separate framework named the “Collection Framework” has been defined in JDK 1.2 which holds all the collection classes and interface in it.

<u>ArrayList Class:</u>
The ArrayList class extends AbstractList and implements the List interface.

Constructors
- ArrayList( )
- ArrayList(Collection<? extends E> c)
- ArrayList(int capacity)

Methods
- void ensureCapacity(int cap)
- void trimToSize( )
- object[ ] toArray( )
- <T> T[ ] toArray(T array[ ])
- boolean add(Object o)
- boolean contains(Object o)
- void add (int index, Object element)
- void addFirst(Object o)
- void addLast(Object o)
- int size()
- boolean remove(Object o)
- int indexOf(Object element)
- int lastIndexOf(Object element)

<u>LinkedList Class:</u>
The LinkedList class extends AbstractSequentialList and implements the List, Deque, and Queue interfaces.

Constructors
- LinkedList( )
- LinkedList(Collection<? extends E> c)

Methods
- boolean add(Object o)
- boolean contains(Object o)
- void add (int index, Object element)
- void addFirst(Object o)
- void addLast(Object o)
- int size()
- boolean remove(Object o)
- int indexOf(Object element)
- int lastIndexOf(Object element)
- addFirst( ) or offerFirst( )

- addLast( ) or offerLast( )
- getFirst( ) or peekFirst( )
- getLast( ) or peekLast( )
- removeFirst( ) or pollFirst( )
- removeLast( ) or pollLast( )

UNIT – III

6. a) Explain the difference between creating a thread by extending Thread class and implementing Runnable interface with an example program. **7M**

The Thread class defines several methods that can be overridden by a derived class. Of these methods, the only one that must be overridden is run( ). This is, of course, the same method required when you implement Runnable. Many Java programmers feel that classes should be extended only when they are being enhanced or modified in some way. So, if you will not be overriding any of Thread’s other methods, it is probably best simply to implement Runnable. Also, by implementing Runnable, your thread class does not need to inherit Thread, making it free to inherit a different class.

Example Program:

```
import java.io.*;
class EvenThread extends Thread{
        public void run(){
                try{
                        PrintWriter out = new PrintWriter("./Even.txt");
                        for(int i = 1; i < 500000; i++){
                                if(i % 2 == 0)
                                        out.println("Even Number:"+i);
                        }
                        out.close();
                }catch(FileNotFoundException e){}
        }
}

class OddThread implements Runnable{
        public void run(){
                for(int i = 1; i < 500000; i++){
                        if(i % 2 == 1)
                                System.out.println("Odd Number:"+i);
                }
        }
}
```

```
class MTDemo{
        public static void main(String[] args){
                long tbe = System.currentTimeMillis();
                Thread et = new Thread(new EvenThread(),"Even Number Printer
                                                                Thread");
                Thread ot = new Thread(new OddThread(),"Odd Number Printer
                                                                Thread");
                et.start();
                ot.start();
                try{
                        et.join();
                        ot.join();
                }catch(InterruptedException e){}
                long tae = System.currentTimeMillis();
                System.out.println("Time to complete execution: "+(tae-tbe));
        }
}
```

Note: Give full marks for any other correct example program.

b) Write a Java program to demonstrate User defined Exceptions. **7M**

Program:

```
class StackOverflowException extends Exception{
        StackOverflowException(String desc){
                super(desc);
        }
}
class StackUnderflowException extends Exception{
        StackUnderflowException(String desc){
                super(desc);
        }
}
class _Stack{
        int stk[];
        int tos, size;
        _Stack(){
                size = 10;
                stk = new int[size];
                tos = -1;
        }
        _Stack(int size){
                this.size = size;
                stk = new int[this.size];
```

```
		tos = -1;
	}
	void push(int ele){
		try{
			if(tos >= size - 1)
				throw new
					StackOverflowException("StackOverflow");
			stk[++tos] = ele;
		}catch(StackOverflowException e){
			System.out.println("Stack Overflow: "+e);
		}
	}
	int pop(){
		try{
			if(tos == -1){
				throw new
				   StackUnderflowException("StackUnderflow");
			}
		}catch(StackUnderflowException e){
			System.out.println("Stack Underflow: "+e);
			System.exit(0);
		}
		return stk[tos--];
	}
	void print(){
		System.out.println("Stack Contents are: ");
		for(int i=tos; i>=0;i--){
			System.out.print(stk[i]+"\t");
		}
		System.out.println();
	}
}
class _StackDemo{
	public static void main(String[] args){
		_Stack s = new _Stack(4);
		for(int i=0;i<10;i++)
			s.push(i+1);
		s.print();
		for(int i=0;i<10;i++)
			System.out.println("Current tos element:"+s.pop());
	}
}
```

Note: Give full marks for any other correct example program.

(OR)

7. a) Write a JAVA program to copy the contents of one file to another file. **7M**

<u>Program:</u>

```
import java.io.*;
class FileCopy{
        public static void main(String[] args) throws IOException{
                if(args.length != 2){
                        System.out.println("Need to pass Two File names as
                                                        command line arguments");
                        System.out.println(">java FileCopy File1 File2");
                        System.exit(-1);
                }
                FileInputStream fis = null;
                FileOutputStream fos = null;
                try{
                        fis = new FileInputStream(args[0]);
                        fos = new FileOutputStream(args[1]);
                }catch(FileNotFoundException e){}
                int ch;
                while((ch = fis.read()) != -1){
                        fos.write((char)ch);
                }
                fis.close();
                fos.close();
        }
}
```

Note: Give full marks for any other correct example program.

b) Explain multi-threading. Write the purpose of isAlive() and join() functions in JAVA. Explain the same with an example. **7M**

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

Sometimes one thread needs to know when other thread is terminating. In java, isAlive() and join() are two different methods that are used to check whether a thread has finished its execution or not.

The isAlive() method returns true if the thread upon which it is called is still running otherwise it returns false. But, join() method is used more commonly than isAlive(). This method waits until the thread on which it is called terminates.

Program:

```
public class MyThread extends Thread {
        public void run() {
                System.out.println("r1 ");
                try {
                        Thread.sleep(500);
                }
                catch(InterruptedException ie) {
                }
                System.out.println("r2 ");
        }
        public static void main(String[] args) {
                MyThread t1=new MyThread();
                MyThread t2=new MyThread();
                t1.start();
                try{
                        t1.join();
                }catch(InterruptedException ie){}
                t2.start();
                System.out.println(t1.isAlive());
                System.out.println(t2.isAlive());
        }
}
```

Note: Give full marks for any other correct example program.

UNIT – IV

8. a) Discuss various AWT containers with examples. 7M

Containers are integral part of AWT GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it adds the capability to add component to itself. Following are noticable points to be considered.

- Sub classes of Container are called as Containter. For example Panel, Frame and Window.
- Container can add only Component to itself.
- A default layout is present in each container which can be overridden using setLayout method.

Container: It is a generic container object which can contain other AWT components.

Commonly used Container Classes:

Panel

Panel is the simplest container. It provides space in which any other component can be placed, including other panels.

Frame

A Frame is a top-level window with a title and a border

Window

A Window object is a top-level window with no borders and no menubar.

b) **Write a short note on the following** **7M**
i) JTextArea ii) JTable

<u>JTextArea:</u>

JTextArea allows editing of multiple lines of text. JTextArea can be used in conjunction with class JScrollPane to achieve scrolling. The underlying JScrollPane can be forced to always or never have either the vertical or horizontal scrollbar.

Creating and Setting or Obtaining Contents

Method or Constructor	Purpose

JTextArea()
JTextArea(String)
JTextArea(String, int, int)
JTextArea(int, int)
	Creates a text area. When present, the String argument contains the initial text. The int arguments specify the desired width in columns and height in rows, respectively.

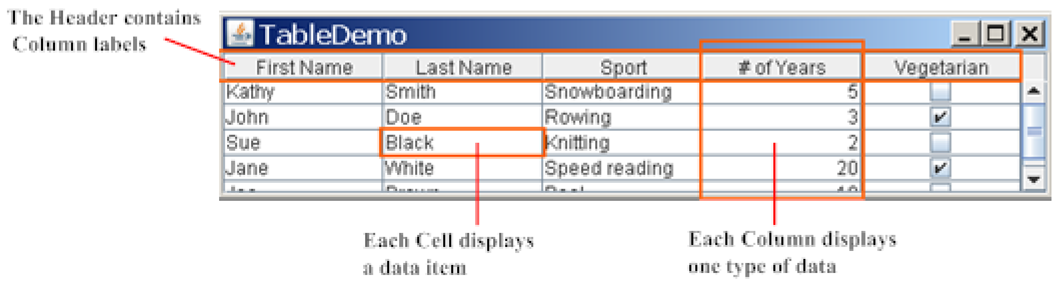
void setText(String) String getText()	Sets or obtains the text displayed by the text area.

Fine Tuning the Text Area's Appearance

Method	Purpose
void setEditable(boolean) boolean isEditable()	Sets or indicates whether the user can edit the text in the text area.
void setColumns(int); int getColumns()	Sets or obtains the number of columns displayed by the text area. This is really just a hint for computing the area's preferred width.
void setRows(int); int getRows()	Sets or obtains the number of rows displayed by the text area. This is a hint for computing the area's preferred height.

<u>JTable:</u>

With the <u>JTable</u> class you can display tables of data, optionally allowing the user to edit the data. JTable does not contain or cache data; it is simply a view of your data. Here is a picture of a typical table displayed within a scroll pane:


The Header contains Column labels
TableDemo
Each Cell displays a data item
Each Column displays one type of data

First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	☐
John	Doe	Rowing	3	✓
Sue	Black	Knitting	2	☐
Jane	White	Speed reading	20	✓

Creating and Setting Up a JTable

Constructor or Method	**Purpose**

JTable()
JTable(int numRows, int numColumns)
JTable(Object[][] data, Object[] colHeads)
	Create a JTable. *data* is a two-dimensional array of the information to be presented, and *colHeads* is a one-dimensional array with the column headings.

void addColumn(String)	Add a column to table.
void addRow(String[])	Add A row to Table.

(OR)

9. a) Explain key events with an example program. **7M**

<u>Program:</u>

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class KeyEventsDemo extends Applet implements KeyListener{
        char ch;
        int x = 250, y = 250;
        public void init(){
                setFocusable(true);
                this.addKeyListener(this);
        }
        public void paint(Graphics g){
                g.drawString("KeyReleased:"+ch,20,20);
                g.fillOval(x,y,30,30);
        }
        public void keyTyped(KeyEvent ke){
                showStatus("Key Typed"+ke.getKeyChar());
        }
        public void keyPressed(KeyEvent ke){
                int keycode = ke.getKeyCode();
                if(keycode == ke.VK_UP){
                        y -= 2;
                        repaint();
                }
                else if(keycode == ke.VK_DOWN){
                        y += 2;
                        repaint();
                }
                else if(keycode == ke.VK_LEFT){
                        x -= 2;
                        repaint();
                }
                else if(keycode == ke.VK_RIGHT){
                        x += 2;
                        repaint();
                }
        }
        public void keyReleased(KeyEvent ke){
                ch=ke.getKeyChar();
                repaint();
        }
}
/*<applet code=KeyEventsDemo height=500 width=500></applet>*/
```

Note: Give full marks for any other correct example program.

b) Write a program to demonstrate any Layout Manager. **7M**

Program:

```
import java.awt.*;
import javax.swing.*;
public class Griddemo extends Frame {
        Griddemo() {
                JButton btn1 = new JButton("Button 1");
                JButton btn2 = new JButton("Button 2");
                JButton btn3 = new JButton("Button 3");
                JButton btn4 = new JButton("Button 4");
                JButton btn5 = new JButton("Button 5");
                JPanel panel = new JPanel(new GridLayout(3, 2, 10, 10));
                panel.add(btn1);
                panel.add(btn2);
                panel.add(btn3);
```

```
		panel.add(btn4);
		panel.add(btn5);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 150);
		add(panel);
		setVisible(true);
	}
	public static void main(String[] args) {
		Griddemo();
	}
}
```
Note: Give full marks for any other correct example program.