SRINIVAS UNIVERSITY

INSTITUTE OF COMPUTER SCIENCE AND INFORMATION SCIENCES

V SEMESTER BCA

Advanced java programming

20BCASD52/20BCAAI52/20BCANT52

QUESTION BANK WITH ANSWERS

UNIT I

UNIT I

Multiple Choice Questions

Questions based on Knowledge

1. Which of these functions is called in a Java program to display the output of an applet?

A. display()
B. paint()
C. displayApplet()
D. appletviewer()
Answer: Option B

2. Which of these methods can be used to output a string in an applet?

A. display()
B. print()
C. drawString()
D. transient()
Answer: Option C

3. What does AWT stands for?

A. All Window Tools
B. All Writing Tools
C. Abstract Window Toolkit
D. Abstract Writing Toolkit
Answer: Option C

4. Which exception is thrown by read() method?

A. IOException
B. InterruptedException
C. SystemException
D. SystemInputException
Answer: Option A

5. Which of these packages contain classes and interfaces used for input & output operations of a program?

A. java.util
B. java.lang
C. java.io
D. All of the mentioned
Answer: Option C

6. Which of these class is not a member class of java.io package?

A. String
B. StringReader
C. Writer
D. File
Answer: Option A

7. Which of these class is not related to input and output stream in terms of functioning?

A. File
B. Writer
C. InputStream
D. Reader
Answer: Option A

8. The DataInputStream and DataOutputStream classes are streams that allow the reading and writing of java primitive data types.

A. file
B. sequence
C. object
D. filter
Answer: Option D

9. The class provides the capacity to read primitive data types from an input stream.
   A. pushbackInputStream
   B. DataInputStream
   C. BufferedInputStream
   D. PipeInputStream

Answer: Option B

10. DataInput is
   A. an abstract class defined in java.io
   B. a class we can use to read primitive data types
   C. an interface that defines methods to open files
   D. an interface that defines methods to read primitives data types

Answer: Option D

11. Which is a special type of program that is embedded in the webpage to generate the dynamic content?
   A. Package
   B. Applet
   C. Browser
   D. None Of The Above

Answer: Option B

12. What is used to run an Applet?
   A. An Html File
   B. An Appletviewer Tool(For Testing Purpose)
   C. Both A & B
   D. None Of The Above

Answer: Option C

13. Which is the correct order of lifecycle in an applet?
   A. Applet Is Started,Initialized,Painted,Destroyed,Stopped
   B. Applet Is Painted,Started,Stopped,Initilaized,Destroyed
   C. Applet Is Initialized,Started,Painted,Stopped,Destroyed
   D. Applet Is Painted,Started, Initilaized, Stopped,Destroyed

Answer: Option C

Questions based on Understanding

1. Which of these is used to perform all input & output operations in Java?
   A. streams
   B. Variables
   C. classes
   D. Methods

Answer: Option A

2. Which of these is a type of stream in Java?
   A. Integer stream
   B. Short stream
   C. Byte stream
   D. Long stream

Answer: Option C

3. Which of these classes are used by Byte streams for input and output operation?
   A. InputStream
   B. InputOutputStream
   C. Reader
   D. All of the mentioned

Answer: Option A

4. Which of these classes are used by character streams for input and output operations?
   A. InputStream
   B. Writer
   C. ReadStream

D. InputOutputStream

Answer: Option B

5. Name the package that contains a large number of stream classes that provide capabilities for processing all types of data.

A. java.awt
B. java.io
C. java.util
D. java.net

Answer: Option B

6. Which of the following method(s) not included in InputStream class.

A. available( )
B. reset( )
C. flush( )
D. close( )

Answer: Option C

7. Which of the following methods not included in OutputStream class.

A. write( )
B. skip( )
C. close( )
D. flush( )

Answer: Option B

8. The following code depicts the creation of a

```
import java.applet.*;
import java.awt.*;
public class main extends Applet
{
public void paint(Graphics g)
{
g.drawString("Welcome in Java Applet.",40,20);
}
}
```

A. Banner Using Applet
B. Basic Applet
C. Display Clock
D. None of The Above

Answer: Option B

9. An applet can play an audio file represented by the AudioClip interface in the java, applet package Causes the audio clip to replay continually in which method?

A. Public Void Play()
B. Public Void Loop()
C. Public Void Stop()
D. None Of The Above

Answer: Option B

10. What invokes immediately after the start() method and also any time the applet needs to repaint itself in the browser?

A. Stop()
B. Init()
C. Paint()
D. Destroy()

Answer: Option C

11. Which method is called only once during the run time of your applet?

A. stop()
B. paint()
C. init()
D. destroy()

Answer: Option C

12. When an applet is terminated which of the following sequence of methods calls take place?

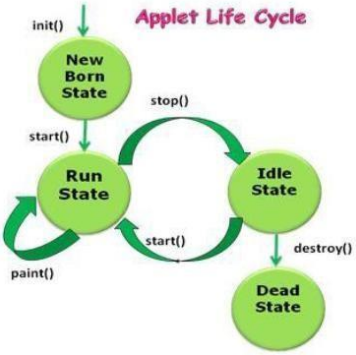
A. stop(),paint(),destroy()
B. destroy(),stop(),paint()
C. destroy(),stop()
D. stop(),destroy()

Answer: Option D

1. **Explain the life cycle of Applet with neat diagram**

When an applet is loaded, it undergoes a series of changes in its state as shown in fig

- Born on initialization state
- Running state
- Idle state
- Dead or destroyed state



Initialization State

Applet enters the initialization state when it is first loaded. This is achieved by calling the init() method of Applet class. The applet is born. At this stage, we may do the following.

- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

```
public void init()
 { ……
   …… }
```

Running State

Applet enters the running state when the system calls the start() method of Applet class. This occurs automatically after the applet is initialized.

```
public voids start()
{ ……
…… }
```

Idle or stopped state

An applet becomes idle when it is stopped from running. Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the stop() method.

```
public void stop()

{……
 …….. }
```

Dead State

An applet is said to be dead when it is removed from memory. This occurs by invoking the destroy() method when we quit the browser. Destroying stage occurs only once in the applet‘s life cycle.

```
public void destroy()
 { ……
……
…… }
```

Display State

Applet moves to the display state whenever it has to perform some output operations on the screen. This happens immediately after the applet enters into the running state. The paint() method is called to accomplish this task.

```
public void paint(Graphics g)
 { ……
 ……..(Display statements)
 }
```

The paint() method is defined in the Applet class.

2. How to build the applet code along with the syntax

Applet code uses the services of two classes, namely, Applet and Graphics from the Java class library. The Applet class which is contained in the java.applet package provides life and behaviour to the applet through its methods such as init(), start() and point(). Java calls the main() method directly to initiate the execution ofthe program . When an applet is loaded, Java automatically calls a series of Applet class methods for starting, running, and stopping the applet code.

The paint() method of the Applet class, when it is called, actually displays the result of the applet code on the screen. The output may be text, graphics, or sound. The paint() method, which requires a Graphics object as an arquement, is defined as follows. public void paint(Graphics g) This requires that the applet code imports the java.awt package that contains the Graphics class. All output operations of an applet are performed using the methods defined in the Graphics class.

Syntax:

```
 Import java.awt.*;
 Importjava.applet.*
 ;
 ……..
 ……..
Public class appletclassname extends Applet
 { ……
Public void paint(Graphics g)
 { ……..
 ……..
 …… …
}
```

. Example:

```
 Import java.awt.*;
 Import java.applet.*;
 Public class HelloJava extends Applet
 {
Public void paint(Graphics g)
  {
 g.drawString(—Hello Java‖, 10,100);
     }
 }
```

3. How to pass the parameters to Applet. Explain with example

We can supply user-defined parameters to an applet using tags. Each tag has a name attribute such as color, and a value attribute such as red. Inside the applet code, the applet can refer to that parameter by name to find its value. For example, we can change the colour of the text displayed to red by an applet by using a < PARAM... > tag as follows.

```
   <APPLET><PARAM = color VALUE = "red‖ >
   </APPLET>
```

Similarly, we can change the text to be displayed by an applet by supplying new text to

the appletthrough a <PARAM >tag as shown below:
<PARAM NAME = text VALUE = ‖I love Java —>

Passing parameters to an applet code using <PARAM>tag is something similar to passing parameters to the main( ) method using command line arguments. To set up and handle parameters, we need to do two things

1. include appropriate tags in the HTML document.
2. Provide Code in the applet to parse these parameters.

Parameters are passed to an applet when it is loaded. We can define the init( ) method in the applet to get hold of the parameters defined in the tags. This is done using the getParameter( ) method., which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

Applet HellojavaParm import java.awt.*; import java.applet.*;public class HelloJavaParam extends Applet

```
{
String str;
public void init( )
{
str = getParameter ("string—); //Receiving parameter value if (str = =
nu1l)str=‖Java‖;
str= —Hello‖ + str //Using the value
}
public void paint (Graphics g)
{
g.drawString(str, 10, 100);
}
}
```

Now we have to create HTML file that contains this applet. below program shows a web page that passesa parameter whose name is "string" and whose VALUE is "Applet!" to the applet HelloJavaParam

The HTML file for HelloJavaParam applet

```
<HTML>
<! Parameterized HTML file>
<HEAD>
<TITLE> Welcome to Java Applets </TITLE>
<HEAD>
<BODY>
<APPLET CODE = HelloJavaParam.class WIDTH = 400
HEIGHT = 200>
<PARAM NAME = "string" VALUE = "Applet! ">
</APPLET>
</BODY>
</HTML>
```

4. Explain about Input/output Exce ption

When creating files and performing i/o operations on them, the system may generate i/o related exception. The basic i/o related exception classes and their functions are given below

EOFException	Signals that an end of the file or end of stream has been reached unexpectedly during input.
FileNotFoundException	Informs that a file could not be found
InterruptedIOException	Warns that an I/O operations has been interrupted
IOException	Signals that an I/O exception of some sort has occurred

Each i/o statement or group of i/o statements must have an exception handler around it as shown below.try

```
{
……
……//I/O statements
}
Catch(IOException e)
{
…….
……}//message output statement
```

5. Explain about Random Access File along with example.

RandomAcessFile class supported by the java.io package allows us to create files that can be used for reading and writing data with random access.

Ie, we can ―jump around‖ in the file while using the file. Such files are known as random access files.A file can be created and opened for random access by giving a mode string as a parameter to the constructor when we open the file.

We can use one of the following two mode strings.

―r‖ for reading only

―rw‖ for both reading and writing

An existing file can be updated using the ―rw‖ mode.

Random access files support a pointer known as file pointer that can be moved to arbitrary positions inthe file prior to reading or writing.

The file pointer is moved using the method **seek()** in the RandomAccessFile class. When the file is opened by the statement

File =new RandomAccessFile("rand. dat");

The file pointer is automatically positioned at the beginning of the file.

Reading/Writing using a random access file

```
Import java.io.*; Class Random
{
Public static void main(String args[])
{
RandomAccessFIle file=null; try
{
file=new RandomAccessFile(―rand.dat‖, ―rw‖);
 file.writeChar(_X‘);
file.writeInt(555);
file.writeDouble(3.1412)
; file.seek(0);
System.out.println(file.readChar());
System.out.println(file.readInt());
System.out.println(file.readDouble());
File.seek(2);
```

```
System.out.println(file.readInt());
file.seek(file. length());
```

```
filewriteBoolean(false);
file.seek(4);
System.out.println(file.readBoolean());
 file.close();
}
Catch(IOException  e)
{
System.out.println(e);
}
}
}
```

6. Write a note on reading and writing bytes.

FileReader and File Writer classes to read and write 16-bit characters. Most file systems use only 8-bitbytes.

Two commonly used classes for handling bytes are FileInputStream and FileOutputStream classes

Writing Bytes to a file

```
import java. io.
*;class
WriteBytes
{
Public static void main(String args[])
{
Byte cities []={_D‘,‘E‘,‘L‘,‘H‘,‘I‘};
FileOutputStream
 outfile=null;try
{
outfile=new FileOutputStream(—c
 ity.txt‖);outfile.write(cities);
outfile.close();
}
catch(IOException e)
{
System.out.println(e
); System.exit(-1);
}
}
}
```

Reading bytes from file

```
import java.io.*;
class ReadBytes
{
Public static void main(String args[])
{
FileInputStream
 infile=null;int b;
try
{
infile=new FileInputStream(
args[0]);while((b=infile.read())!=-
1)
{
System.out.println((char)b);
```

```
}
infile.close();
}
catch(IOException e)
```

```
{
System.out.println(e);
}
}
}
```

7. Write a note on reading and writing characters

The two subclasses used for handling characters in files are FileReader(for reading characters) and FileWriter(for writing characters).

The below program uses these two file stream classes to copy the contents of a file named — input.dat‖ into a file called —output.dat‖.

```
Import java.io.*; Class CopyCharacters
{
Public static void main(String args[])
{
//Declare and create input and output
filesFile infile=new File(—input.dat‖);
File outfile=new File(—output.dat‖);
FileReader ins=null; //Creates file stream ins
FileWriter outs=null; //Creates file stream outs
try
{
ins=new FileReader(infile); //opens infile

outs=new FileWriter(outFile); //Opens outfile
//Read and write till the end int
ch;While((ch=ins.read())!=-1)
{
Outs.write(ch);
}
}
Catch(IOException e)
{
System.out.println(e
); System.exit(-1);
}
Finally //close files
{
try
{
Ins.close();
Outs.close();
}
Catch(IOException e){}
}
}
}
```

8. What are the proce dures to create the file.

If we want to create and use a disk file,we need to decide the follow ing about the file and its intendedpurpose.

- Suitable name for file
- the Data type to be
- stored

Purpose(reading,writing, or updating)

- Method of creating the file

A filename is a unique string of characters that helps identify a file on the disk. Example: input.data orinput.txt
Data type is to decide the type of file stream classes to be used for handling the data. We should decide whether the data to be handled is in the form of characters, bytes or primitive type.
The purpose of using a file must also be decided before using it. For example, we should know whetherthe file is created for reading only, or writing only, or both the operations.

For using a file, it must be opened first. This is done by creating a file stream and then linking it to the filename. A file stream can be defined using the classes of **Reader/InputStream** for reading data and **Writer/OutputStream** for writing data.

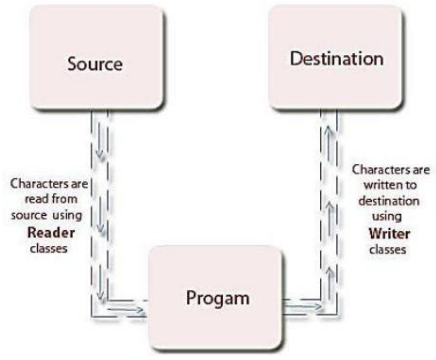
The java.io package in cludes a class known as the **File** class that provides support for creating files anddirectories. The class includes several constructing for instantiating the File objects. This class also contains several methods for supporting the operations such as

- Creating a file
- Opening a file
- Closing a file
- Deleting a file
- Getting the name of a file
- Getting the size of a file
- Checking the existence of a file
- Renaming a file
- Checking whether the file is writable
- Checking whether the file is readable

9. Explain about character stream classes.

Character Stream Classes are used to read characters from the source and write **characters** to destination. There are two kinds of Character Stream classes - Reader classes and Writer classes.

- **Reader Classes** - These classes are subclasses of an abstract class, Reader and they are used to read characters from a source(file, memory or console).
- **Writer Classes** - These classes are subclasses of an abstract class, Writer and they used to writecharacters to a destination(file, memory or console).\



Reader

Reader class and its subclasses are used to read characters from source. Reader class is a base class of all the classes that are used to read characters from a file memory or console. Reader is an abstract class and hence we can't instantiate it but we can use its subclasses for reading characters from the input stream.
We will discuss subclasses of Reader class with their code, in the next few articles

Methods of Reader classes

These methods are of Reader class

Methods	Description
Int read()	This method reads a character from the input stream
Int read(char[], ch)	This method reads a chunk of characters from the input stream and store them in its char array, ch.
close()	This method closes this output stream and also frees any system resources connected with it.

Writer

Writer class and its subclasses are used to write characters to a file, memory or console. Writer is an abstract class and hence we can't create its object but we can use its subclasses for writing characters to the output stream. We will discuss subclasses of Writer with their code in the next few artic les

Methods of writer classes

Methods of Writer class provide support for writing characters to the output stream. As this is an abstractclass. Hence, some undefined abstract methods are defined in ths subclasses of Output Stream

Methods	Description
abstract void flush()	This method flushes the output steam by forcing out buffered bytes to be written out.
void write(int c)	This method writes a characters(contained in an int) to the output stream.
void write(char[] arr)	This method writes a whole char array(arr) to the output stream.
abstract void close()	This method closes this output stream and also frees any resources connected with this output stream.

10. Write all the methods of InputStream and OutputStream classes

The below table gives a brief description of all the methods provided by the **InputStream** class

Method Description

1. Read() Reads a byte from the input stream
2. Read(byte b[]) Reads an array of bytes into b
3. Read(byte b[], int n, int m) Reads m bytes into b starting from nth byte
4. Available() Gives number of bytes availale in the input
5. Skip(n) skips over n bytes from the input stream
6. Reset() Goes back to the beginning of the stream
7. Close closes the input stream

Table gives a brief description of all the methods of outputstream class\

Method Description

1. Write() Writes a byte to the output stream
2. write(byte[] b) Writes m bytes in the array b to the output stream
3. write(byte b[], int n, int m) Writes m bytes from array b starting from nth byte
4. close() closes the output stream
5. flush() flushes the output stream

UNIT II

Multiple Choice Questions

Questions based on Understanding

1. Which is the container that doesn't contain title bar and MenuBars but it can have other components like button, textfield etc?

A. Window B. Frame C. Panel D. Container

Answer: Option C

2. Which package provides many event classes and Listener interfaces for event handling?

A. java.awt B. java.awt.Graphics

C. java.awt.event D. java.event

Answer: Option C

3. In Graphics class which method is used to draws a rectangle with the specified width and height?

A. public void drawRect(int x, int y, int width, int height)

B. public abstract void fillRect(int x, int y, int width, int height)

C. public abstract void drawLine(int x1, int y1, int x2, int y2)

D. public abstract void drawOval(int x, int y, int width, int height)

Answer: Option A

4. Name the class used to represent a GUI application window, which is optionally resizable and can have a title bar, an icon, and menus.

A. Window B. Panel C. Dialog D. Frame

Answer: Option D

5. To use the ActionListener interface it must be implemented by a class there are several ways to do that find in the following?

A. Creating a new class B. Using the class the graphical component

C. An anonymous inner class D. All mentioned above

Answer: Option D

6. Which is a component in AWT that can contain another component like buttons, text fields, labels etc.?

A. Window B. Container C. Panel D. Frame

Answer: Option B

7. Which is used to store data and partial results, as well as to perform dynamic linking, return values for methods, and dispatch exceptions?

A. Window B. Panel C. Frame D. Container

Answer: Option C

8. What are the different types of controls in AWT?

A. Pushbuttons B. Checkboxes C. Choice lists D. All of these

Answer: Option D

9. Which class provides many methods for graphics programming?

A. java.awt B. java.Graphics

C. java.awt.Graphics D. java. awt.event

Answer: Option C

10. By which method You can set or change the text in a Label?

A. setText() B. getText() C. Both A & B D. setLabel

Answer: Option A

11. Which class can be used to represent a checkbox with a textual label that can appear in a menu.

A. MenuBar B. MenuItem C. CheckboxMenuItem D.Menu

Answer: Option C

12. How many types of controls does AWT supports these controls are subclasses of component?

A. 7 B. 6 C. 5 D. 8

Answer: Option A

13. Which are passive controls that do not support any interaction with the user?

A. Choice B. List C. Labels D. Checkbox

Answer: Option C

14. Which of the following classes are derived from the Component class?
A. Container B. Window C. List D.MenuItem
Answer: Option D
15. How many ways can we align the label in a container?
A. 1 B. 2 C. 3 D. 4
Answer: Option C
16. Which method is used to set the graphics current color to the specified color in the graphics class?
A. public abstract void setFont(Font font)
B. public abstract void setColor(Color c)
C. public abstract void drawString(String str, int x, int y)
D. public abstract void getColor(Color c)
Answer: Option B
17. Which object can be constructed to show any number of choices in the visible window?
A. Labels
B. Choice
C. List
D. Checkbox
Answer: Option C

18. Which class is used for this Processing Method processActionEvent( )?
A. Button,List,MenuItem
B. Button,Checkbox,Choice
C. Scrollbar,Component,Button
D. None of the above
Answer: Option A
19. Which package provides many event classes and Listener interfaces for event handling?
A. java.awt
B. java.awt.Graphics
C. java.awt.event
D. None of the above
Answer: Option C
20. Name the class used to represent a GUI application window, which is optionally resizable and can have a title bar, an icon, and menus.
A. Window
B. Panel
C. Dialog
D. Frame Answer:
Option D
21. What does the following line of code do?
Textfield text = new Textfield(10);
A. Creates text object that can hold 10 rows of text.
B. Creates the object text and initializes it with the value 10.
C. The code is illegal.
D. Creates text object that can hold 10 columns of text.
Answer: Option D
22. Which of the following methods can be used to remove a java.awt.Component object from the display?
A. delete()
B. remove()
C. disappear()
D. hide()
Answer: Option D
23. The setBackground() method is part of the following class in java.awt package:
A. Component

B. Graphics
C. Applet
D. Container
Answer: Option A

24. When we invoke **repaint()** for a java.awt.Component object, the AWT invokes the method:
A. update()
B. draw()
C. show()
D. paint()

Answer: Option A

25. Where are the following four methods commonly used?
1) public void add(Component c)
2) public void setSize(int width,int height)
3) public void setLayout(LayoutManager m)
4) public void setVisible(boolean)

A. Graphics class
B. Component class
C. Both A & B
D. None of the above

Answer: Option B

Long Answers

1. Illustrate the use of Label with suitable example.

=>A label is an object of type Label, and it contains a string,
which it displays. Labels are passive controls that do not support any interaction with the user.
Label defines the following constructors:

Label( ) throws HeadlessException
Label(String str) throws HeadlessException
Label(String str, int how) throws HeadlessException
The first version creates a blank label. The second version creates a label that contains the string specified by str. This string is left-justified. The third version creates a label that contains the string specified by str using the alignment specified by how. The value of how must be one of these three constants: Label.LEFT, Label.RIGHT, or Label.CENTER. You can set or change the text in a label by using the setText( ) method. You can obtain the current label by calling getText( ).

```
// Demonstrate Labels
import java.awt.*;
import java.applet.*;
/*
<applet code="LabelDemo" width=300 height=200>
</applet>
*/
public class LabelDemo extends Applet {
public void init( ) {
Label one = new Label("One");
Label two = new Label("Two");
Label three = new Label("Three");
// add labels to applet window
add(one);
add(two);
add(three);
}
}
```

The labels are organized in the window by the default layout manager.

2.List and explain various methods associated with Button.

=>A push button is a component that
contains a label and that generates an event when it is pressed. Push buttons are objects of type Button. Button defines these two constructors:
Button( ) throws HeadlessException
Button(String str) throws HeadlessException
The first version creates an empty button. The second creates a button that contains str as a label. After a button has been created, you can set its label by calling setLabel( ). You can retrieve its label by calling getLabel( ).
Handling Buttons: Each time a button is pressed, an action event is generated. This is sent to any listeners that previously registered an interest in receiving action event notifications from that component. Each listener implements the ActionListener interface. That interface defines the actionPerformed( ) method, which is called when an event occurs. An ActionEvent object is supplied as the argument to this method. It contains both a reference to the button that generated the event and a reference to the action command string associated with the button.

By default, the action command string is the label of the button. Usually, either the button reference or the action command string can be used to identify the button.

3.Explain the purpose of Choice controls with an example.

The Choice class is used to create a pop-up list of items from which the user may choose. Thus,a Choice control is a form of menu. When inactive, a Choice component takes up only enough space to show the currently selected item. When the user clicks on it, the whole list of choices pops up, and a new selection can be made. Each item in the list is a string that appears as a left-justified label in the order it is added to the Choice object. Choice only defines the default constructor, which creates an empty list. To add a selection to the list, call add( ). It has this general form:
void add(String name)
To determine which item is currently selected, you may call either getSelectedItem( ) or getSelectedIndex( ). These methods are shown here:
String getSelectedItem( )
int getSelectedIndex( )
The getSelectedItem( ) method returns a string containing the name of the item. getSelectedIndex( ) returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.
To obtain the number of items in the list, call getItemCount( ). You can set the currently selected item using the select( ) method with either a zero-based integer index or a string that will match a name in the list.

4.Explain List control with an example.

The List class provides a compact, multiple-choice, scrolling selection list. Unlike the Choice object, which shows only the single selected item in the menu, a List object can be constructed to show any number of choices in the visible window. It can also be created to allow multiple selections. List provides these constructors:
List( ) throws HeadlessException
List(int numRows) throws HeadlessException
List(int numRows, boolean multipleSelect) throws HeadlessException
The first version creates a List control that allows only one item to be selected at any one time. In the second form, the value of numRows specifies the number of entries in the list that will always be visible (others can be scrolled into view as needed). In the third form, ifmultipleSelect is true, then the user may select two or more items at a time. If it is false, then only one item may be selected. To add a selection to the list, call add( ). It has the following two forms:
void add(String name)
void add(String name, int index)
Here, name is the name of the item added to the list. The first form adds items to the end of the list. The second form adds the item at the index specified by index. Indexing begins at zero. You can specify –1 to add the item to the end of the list. For lists that allow only single selection, you can determine which item is currently selected by calling either getSelectedItem( ) or getSelectedIndex( ). These methods are shown here:
String getSelectedItem( )
int getSelectedIndex( )
The getSelectedItem( ) method returns a string containing the name of the item. If more than one item is selected, or if no selection has yet been made, null is returned. getSelectedIndex( )

returns the index of the item.
int getItemCount( )
void select(int index)
Given an index, you can obtain the name associated with the item at that index by calling getItem( ), which has this general form:
String getItem(int index)
Here, index specifies the index of the desired item.
To process list events, you will need to implement the ActionListener interface. Each time a List item is double-clicked, an ActionEvent object is generated. Its getActionCommand( ) method can be used to retrieve the name of the newly selected item. Also, each time an item is selected or deselected with a single click, an ItemEvent object is generated. Its getStateChange( )
method can be used to determine whether a selection or deselection triggered this event. getItemSelectable( ) returns a reference to the object that triggered this event. Here is an example that converts the Choice controls in the preceding section into List components, one multiple choice and the other single choice

5. Explain Scroll Bars.

Scroll bars are used to select continuous values between a specified minimum and maximum. Scroll bars may be oriented horizontally or vertically. A scroll bar is actually a composite of several individual parts. Each end has an arrow that you can click to move the current value of the scroll bar one unit in the direction of the arrow.
Scrollbar( ) throws HeadlessException
Scrollbar(int style) throws HeadlessException
Scrollbar(int style, int initialValue, int thumbSize, int min, int max)
throws HeadlessException
The first form creates a vertical scroll bar. The second and third forms allow you to specify the orientation of the scroll bar. If style is Scrollbar.VERTICAL, a vertical scroll bar is created. If style is Scrollbar.HORIZONTAL, the scroll bar is horizontal. In the third form of the constructor, the initial value of the scroll bar is passed in initialValue. The number of units represented by the height of the thumb is passed in thumbSize. The minimum and maximum values for the scroll bar are specified by min and max.
If you construct a scroll bar by using one of the first two constructors, then you need to set its parameters by using setValues( ), shown here, before it can be used:
void setValues(int initialValue, int thumbSize, int min, int max)
The parameters have the same meaning as they have in the third constructor just described. To obtain the current value of the scroll bar, call getValue( ). It returns the current setting. To set the current value, call setValue( ). These methods are as follows:
int getValue( )
void setValue(int newValue)
Here, newValue specifies the new value for the scroll bar. When you set a value, the slider box inside the scroll bar will be positioned to reflect the new value. You can also retrieve the minimum and maximum values via getMinimum( ) and getMaximum( ), shown here:
int getMinimum( )
int getMaximum( )
They return the requested quantity. By default, 1 is the increment added to or subtracted from the scroll bar each time it is scrolled up or down one line. You can change this increment by calling setUnitIncrement( ).

6. Explain about Layout Managers.

Layout manager automatically arranges controls within a window by using some type of algorithm. If you have programmed for other GUI environments, such as Windows, then you are accustomed to laying out your controls by hand. While it is possible to lay out Java controls by hand, too,you generally won‘t want to, for two main reasons. First, it is very tedious to manually layout a large number of components. Second, sometimes the width and height information is not yet available when you need to arrange some control, because the native toolkit components haven‘t been realized.
Each Container object has a layout manager associated with it. A layout manager is an instance of any class that implements the LayoutManager interface. The layout manager is set by the setLayout( ) method. If no call to setLayout( ) is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The setLayout( ) method has the following general form:
void setLayout(LayoutManager layoutObj)
Here, layoutObj is a reference to the desired layout manager. If you wish to disable the layout manager and position components manually, pass null for layoutObj. If you do this, you will need to determine the shape and position of each component manually, using the setBounds( ) method defined by Component. Normally, you will want to use a layout mana**ger.**

7. With an example explain FlowLayout.

FlowLayout is the default layout manager. FlowLayout implements a simple layout style, which is similar to how words flow in a text editor. The direction of the layout is governed by the container‘s component orientation property, which, by default, is left to right, top to bottom. Therefore, by default, components are laid out line-by-line beginning at the upper- left corner. In all cases, when a line is filled, layout advances to the next line. A small space is left between each component, above and below, as well as left and right. Here are the constructors for FlowLayout:
FlowLayout( )
FlowLayout(int how)
FlowLayout(int how, int horz, int vert)
The first form creates the default layout, which centers components and leaves five pixels of space between each component. The second form lets you specify how each line is aligned. Valid values for how are as follows:
FlowLayout.LEFT
FlowLayout.CENTER
FlowLayout.RIGHT
FlowLayout.LEADINGFlowLayout.TRAILING
These values specify left, center, right, leading edge, and trailing edge alignment, respectively. The third constructor allows you to specify the horizontal and vertical space left between components in horz and vert, respectively. Here is a version of the CheckboxDemo applet shown earlier in this chapter, modified so that it uses left-aligned flow layout:

```
// Use left-aligned flow layout.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="FlowLayoutDemo" width=250 height=200>
```

```
</applet>
*/
public class FlowLayoutDemo extends Applet
implements ItemListener {
String msg = "";
Checkbox winXP, winVista, solaris, mac;
public void init( ) {
// set left-aligned flow layout
setLayout(new FlowLayout(FlowLayout.LEFT));
winXP = new Checkbox("Windows XP", null, true);
winVista = new Checkbox("Windows Vista");
solaris = new Checkbox("Solaris");
mac = new Checkbox("Mac OS");
add(winXP);
add(winVista);
add(solaris);
add(mac);
// register to receive item events
winXP.addItemListener(this);
winVista.addItemListener(this);
solaris.addItemListener(this);
mac.addItemListener(this);
}
// Repaint when status of a check box changes.
public void itemStateChanged(ItemEvent ie) {
repaint();
}
// Display current state of the check boxes.
public void paint(Graphics g) {
msg = "Current state: ";
g.drawString(msg, 6, 80);
msg = " Windows XP: " + winXP.getState();
g.drawString(msg, 6, 100);
msg = " Windows Vista: " + winVista.getState();
g.drawString(msg, 6, 120);
msg = " Solaris: " + solaris.getState();
g.drawString(msg, 6, 140);
msg = " Mac: " + mac.getState();g.drawString(msg, 6, 160);
}
}
```

8. With an example explain BorderLayout.

BorderLayout()
BorderLayout(int horz, int vert)
The first form creates a default border layout. The second allows you to specify the horizontal and vertical space left between components in horz and vert, respectively. BorderLayout defines constants that specify the regions as BorderLayout.CENTER, BorderLayout.SOUTH, BorderLayout.EAST, BorderLayout.WEST, BorderLayout.NORTH.
When adding components, you will use these constants with the following form of add( ),

which is defined by Container:
void add(Component compObj, Object region)
Here, compObj is the component to be added, and region specifies where the component will be added. Here is an example of a BorderLayout with a component in each layout area:

```
// Demonstrate BorderLayout.
import java.awt.*;
import java.applet.*;
import java.util.*;
/*
<applet code="BorderLayoutDemo" width=400 height=200>
</applet>
*/
public class BorderLayoutDemo extends Applet {
public void init( ) {
setLayout(new BorderLayout());
add(new Button("This is across the top."),
BorderLayout.NORTH);
add(new Label("The footer message might go here."),
BorderLayout.SOUTH);
add(new Button("Right"), BorderLayout.EAST);
add(new Button("Left"), BorderLayout.WEST);
String msg = "The reasonable man adapts " + "himself to the world;\n" +
"the unreasonable one persists in " +
"trying to adapt the world to himself.\n" +
"Therefore all progress depends " +
"on the unreasonable man.\n\n" + " - George Bernard Shaw\n\n";
add(new TextArea(msg), BorderLayout.CENTER);
}
}
```

9. What is the purpose of layout managers? With an example explain the use of Grid Layout.

Layout manager automatically arranges controls within a window by using some type of algorithm. If you have programmed for other GUI environments, such as Windows, then you are accustomed to laying out your controls by hand. While it is possible to lay out Java controls by hand, too,you generally won‘t want to, for two main reasons. First, it is very tedious to manually layout a large number of components. Second, sometimes the width and height information is not yet available when you need to arrange some control, because the native toolkit components haven‘t been realized.
Each Container object has a layout manager associated with it. A layout manager is an instance of any class that implements the LayoutManager interface. The layout manager is set by the setLayout( ) method. If no call to setLayout( ) is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The setLayout( ) method has the following general form:
void setLayout(LayoutManager layoutObj)

GridLayout
GridLayout lays out components in a two-dimensional grid. When you instantiate a

GridLayout, you define the number of rows and columns. The constructors supported by GridLayout are shown here:
GridLayout( )
GridLayout(int numRows, int numColumns)
GridLayout(int numRows, int numColumns, int horz, int vert)
The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the horizontal and vertical space left between components in horz and vert, respectively. Either numRows or numColumns can be zero. Specifying numRows as zero allows for unlimited- length columns. Specifying numColumns as zero allows for unlimited-length rows. Here is a sample program that creates a 4×4 grid and fills it in with 15 buttons, each labeled with its index:

```
// Demonstrate GridLayout
import java.awt.*;
import java.applet.*;
/*
<applet code="GridLayoutDemo" width=300 height=200>
</applet>
*/
public class GridLayoutDemo extends Applet {
static final int n = 4;
public void init( ) {
setLayout(new GridLayout(n, n));
setFont(new Font("SansSerif", Font.BOLD, 24));
for(int i = 0; i < n; i++) {
for(int j = 0; j < n; j++) {
int k = i * n + j;
if(k > 0)
add(new Button("" + k));
}
}
}
}
```

10. Name the controls required for creating a menu. With an example explain the creation of a menu.

=>A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. Each choice is associated with a drop-down menu. This concept is implemented in the AWT by the following classes:MenuBar,Menu, andMenuItem. In general, a menu bar contains one or more Menu objects. EachMenu object contains a list of MenuItem objects. Each MenuItem object represents something that can be selected by the user. SinceMenu is a subclass ofMenuItem, a hierarchy of nested submenus can be created. It is also possible to include checkable menu items. These are menu options of type CheckboxMenuItem and will have a check mark next to them when they are selected. To create a menu bar, first create an instance ofMenuBar. This class only defines the defaultconstructor. Next, create instances ofMenu that will define the selections displayed on the
bar. Following are the constructors for Menu:
Menu( ) throws HeadlessException
Menu(String optionName) throws HeadlessException
Menu(String optionName, boolean removable) throws HeadlessException
Here, optionName specifies the name of the menu selection. If removable is true, the menu can be removed and allowed to float free. Otherwise, it will remain attached to the menu bar.

(Removable menus are implementation-dependent.) The first form creates an empty menu. Individual menu items are of typeMenuItem. It defines these constructors:
MenuItem( ) throws HeadlessException
MenuItem(String itemName) throws HeadlessException
MenuItem(String itemName, MenuShortcut keyAccel) throws HeadlessException
Here, itemName is the name shown in the menu, and keyAccel is the menu shortcut for this item. You can disable or enable a menu item by using the setEnabled( ) method.

11. List and explain different methods associated with check boxes.

Here, newName becomes the new name of the invoking menu item. getLabel( ) returns the current name. You can create a checkable menu item by using a subclass of MenuItem called CheckboxMenuItem. It has these constructors:
CheckboxMenuItem( ) throws HeadlessException
CheckboxMenuItem(String itemName) throws HeadlessException
CheckboxMenuItem(String itemName, boolean on) throws HeadlessException
Here, itemName is the name shown in the menu. Checkable items operate as toggles. Each time one is selected, its state changes. In the first two forms, the checkable entry is unchecked. In the third form, if on is true, the checkable entry is initially checked. Otherwise, it is cleared. You can obtain the status of a checkable item by calling getState( ). You can set it to a known state by using setState( ). These methods are shown here:
boolean getState( )
void setState(boolean checked)

12. What is the purpose of TextField class? Explain any three methods of TextField class with syntax and example.

=>The TextField class implements a single-line text-entry area, usually called an edit control. Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys, and mouse selections. TextField is a subclass of TextComponent. TextField defines the following constructors:TextField( ) throws HeadlessException
TextField(int numChars) throws HeadlessException
TextField(String str) throws HeadlessException
TextField(String str, int numChars) throws HeadlessException
The first version creates a default text field. The second form creates a text field that is numChars characters wide. The third form initializes the text field with the string contained in str. The fourth form initializes a text field and sets its width.
TextField (and its superclass TextComponent) provides several methods that allow you to utilize a text field. To obtain the string currently contained in the text field, call getText( ). To set the text, call setText( ). These methods are as follows:
String getText( )
void setText(String str)
Here, str is the new string. The user can select a portion of the text in a text field. Also, you can select a portion of text under program control by using select( ). Your program can obtain the currently selected text by calling getSelectedText( ). These methods are shown here:
String getSelectedText( )
void select(int startIndex, int endIndex)
getSelectedText( ) returns the selected text. The select( ) method selects the characters beginning at startIndex and ending at endIndex–1.
You can control whether the contents of a text field may be modified by the user by calling setEditable( ). You can determine editability by calling isEditable( ). These methods are shown

here:
boolean isEditable( )
void setEditable(boolean canEdit)
isEditable( ) returns true if the text may be changed and false if not. In setEditable( ), if canEdit is true, the text may be changed. If it is false, the text cannot be altered. There may be times when you will want the user to enter text that is not displayed, such as a password. You can disable the echoing of the characters as they are typed by calling setEchoChar( ). This method specifies a single character that the TextField will display when characters are entered (thus, the actual characters typed will not be shown). You can check a text field to see if it is in this mode with the echoCharIsSet( ) method. You can retrieve the echo character by calling the getEchoChar( ) method. These methods are as follows:
void setEchoChar(char ch)
boolean echoCharIsSet( )
char getEchoChar( )
Here, ch specifies the character to be echoed. Since text fields perform their own editing functions, your program generally will not respond to individual key events that occur within a text field. However, you may want to respond when the user presses ENTER. When this occurs, an action event is generated. Here is an example that creates the classic user name and password screen:

```
// Demonstrate text field.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="TextFieldDemo" width=380 height=150>
</applet>
*/
public class TextFieldDemo extends Applet
implements ActionListener {
TextField name, pass;
public void init() {
Label namep = new Label("Name: ", Label.RIGHT);
Label passp = new Label("Password: ", Label.RIGHT);
name = new TextField(12);
pass = new TextField(8);
pass.setEchoChar('?');
add(namep);
add(name);
add(passp);
add(pass);
// register to receive action events
name.addActionListener(this);
pass.addActionListener(this);
}
// User pressed Enter.
public void actionPerformed(ActionEvent ae) {
repaint( );
}
public void paint(Graphics g) {
g.drawString("Name: " + name.getText(), 6, 60);
g.drawString("Selected text in name: " + name.getSelectedText(), 6, 80);
g.drawString("Password: " + pass.getText(), 6, 100);
```

```
}
}
```

UNIT III

Multiple Choice Questions

Questions based on Application

1. Where are the following four methods commonly used?

a) public void add(Component c)
b) public void setSize(int width,int height)
c) public void setLayout(LayoutManager m)
d) public void setVisible(boolean)

A. Graphics class
B. Component class
C. Both a & b
D. None of the above
Answer: Option B

2. Name the container that cannot contain title bar and MenuBars but it can have other components like button, textfield

A. Window
B. Frame
C. Panel
D. Container
Answer: Option C

3. Which object can be constructed to show any number of choices in the visible window?

A. Labels
B. Choice
C. List
D. Checkbox
Answer: Option C

4. Object which can store group of other objects is called

A. Collection object
B. Java object
C. Package
D. Wrapper Answer:
Option A

5. JFrame myFrame = new JFrame (); The above statement is called

A. Constructor
B. Layout manager
C. Parameter
D. GUI
Answer: Option A

6. What are the names of the list layout managers in Java?

A. Flow Layout Manager
B. Grid Layout Manager
C. Box Layout Manager
D. All of the above
Answer: Option D

7. What is the name of the Swing class that is used for frames?

A. Window
B. Frame
C. JFrame
D. SwingFrame
Answer: Option C

8. What is the name of the Swing class that is used for frames?

A. swing
B. Frame
C. applet
D. SwingFrame

Answer: Option A

9. Which method is used to set the text of a Label object?

A. setText( )
B. setLabel( )
C. setTextLabel( )
D. setLabelText( ) Answer: Option A

10. The layout of a container can be altered by using which of the following methods.

A. setLayout(aLayoutManager)
B. layout(aLayoutManager)
C. addLayout(aLayoutManager)
D. setLayoutManager(aLayoutManager) Answer: Option A

11. Which of these methods can be used to know which key is pressed?

A. getActionEvent()
B. getActionKey()
C. getModifier()
D. getKey() Answer: Option C

12. Which of these event is generated when a button is pressed?

A. WindowEvent
B. ActionEvent
C. WindowEvent
D. KeyEvent Answer: Option B

13. Which of these packages contains all the classes and methods required for event handling in java?

A. Java.applet
B. Java.awt
C. Java.awt.event
D. Java.event

Answer: Option C

14. In Java, what do you call an area on the screen that has nice borders and various buttons along the top border?

A. A window
B. A screen
C. A box
D. A frame Answer: Option D

15. Suppose you are developing a Java Swing application and want to toggle between various views of the design area. Which of the views given below are present for the users to toggle?

A. Design View
B. Requirements View
C. Source View
D. Management View Answer: Option B

16. The size of a frame on the screen is measured in:

A. Inches
B. centimetres
C. Dots
D. Pixels

Answer: Option D

17. The following statements are the advantages of which of the following Java concepts.

It is lightweight.
It supports pluggable look and feel.

It follows MVC (Model View Controller) architecture.
A. Swing
B. AWT
C. Applets
D. All the above
Answer: Option A
18. Which class provides many methods for graphics programming?
A. java.awt
B. java.Graphics
C. java.awt.Graphics
D) None of the above
Answer: Option C
19. The ActionListener interface is used for handling action events,
A) JButton
B) JCheckbox
C) JMenuItem
D) All of these
Answer: Option D
19. Which class is used to create a pop-up list of items from which the user may choose?
A. List
B. Choice
C. Labels
D. Checkbox
Answer: Option B
20. Which class is used for this Processing Method processActionEvent( )?
A. Button,List,MenuItem
B. Button,Checkbox,Choice
C. Scrollbar,Component,Button
D. None of the above
Answer: Option A
21. The Swing Component classes that are used in Encapsulates a mutually exclusive set of buttons?
A. AbstractButton
B. ButtonGroup
C. JButton
D. ImageIcon Answer:
Option B
22. Swing Components are _____
A. HeavyWeight
B. LightWeight
C. Visually unappealing
D. Do not support plug and feel affect
Answer: Option B
23. A _____ is an independent visual control in Java Swings
A. Container
B. Component
C. AWT
D. Panel
Answer : option B
24. Swing components are derived from _____class
A. JSwing
B. JContainer
C. JComponent
D. JPanel
Answer: option C

25. Which of the following is NOT a top level container in Swings?
A. JFrame
B. JApplet
C. JWindow
D. JPanel
Answer: Option D

Long Answers

1. Explain the advantages of Swing.
Ans.
Two Key Swing Features: Swing was created to address the limitations present in the AWT. It does this through two key features: lightweight components and a pluggable look and feel. Together they provide an elegant, yet easy-to-use solution to the problems of the AWT. More than anything else, it is these two features that define the essence of Swing.
Swing Components Are Lightweight: With very few exceptions, Swing components are lightweight. This means that they are written entirely in Java and do not map directly to platform-specific peers. Because lightweight components are rendered using graphics primitives, they can be transparent, which enables nonrectangular shapes. Thus, lightweight components are more efficient and more flexible. Furthermore, because lightweight components do not translate into native peers, the look and feel of each component is determined by Swing, not by the underlying operating system. This means that each component will work in a consistent manner across all platforms.
Swing Supports a Pluggable Look and Feel: Swing supports a pluggable look and feel (PLAF). Because each Swing component is rendered by Java code rather than by native peers, the look and feel of a component is under the control of Swing. This fact means that it is possible to separate the look and feel of a component from the logic of the component, and this is what Swing does. Separating out the look and feel provides a significant advantage: it becomes possible to change the way that a component is rendered without affecting any of its other aspects. In other words, it is possible to ―plug in‖ a new look and feel for any given component without creating any side effects in the code that uses that component.

2. Write a short note on containers.
Ans.
Containers
Swing defines two types of containers. The first are top-level containers: JFrame, JApplet, JWindow, and JDialog. These containers do not inherit JComponent. They do, however, inherit the AWT classes Component and Container. Unlike Swing‘s other components, which are lightweight, the top-level containers are heavyweight. This makes the top-level containers a special case in the Swing component library.
As the name implies, a top-level container must be at the top of a containment hierarchy. A top-level container is not contained within any other container. Furthermore, every containment hierarchy must begin with a top-level container. The one most commonly used for applications is JFrame. The one used for applets is JApplet. The second type of containers supported by Swing are lightweight containers. Lightweight containers do inherit JComponent. An example of a lightweight container is JPanel, which is a general-purpose container. Lightweight containers are often used to organize and manage groups of related components because a lightweight container can be contained within another container. Thus, you can use lightweight containers such as JPanel to create subgroups of related controls that are contained within an outer container..

3. Briefly explain components.
Ans.
Components: In general, Swing components are derived from the JComponent class.

JComponent provides the functionality that is common to all components. For example, JComponent supports the pluggable look and feel. JComponent inherits the AWT classes Container and Component. Thus, a Swing component is built on and compatible with an

AWT component. All of Swing‘s components are represented by classes defined within the package javax.swing. The following are the class names for Swing components.
JApplet, JButton, JCheckBox, JCheckBoxMenuItem, JColorChooser, JComboBox, JComponent, JDesktopPane, JDialog, JEditorPane, JFileChooser, JFormattedTextField, JFrame, JInternalFrame, JLabel, JLayeredPane, JList JMenu, JMenuBar and JMenuItem etc.
Notice that all component classes begin with the letter J. For example, the class for a label is JLabel; the class for a push button is JButton; and the class for a scroll bar is JScrollBar.

4. Mention the purpose of different layout managers available in Swing
Ans.
Layout manager automatically arranges controls within a window by using some type of algorithm. If you have programmed for other GUI environments, such as Windows, then you are accustomed to laying out your controls by hand. While it is possible to lay out Java controls by hand, too,you generally won‘t want to, for two main reasons. First, it is very tedious to manually layout a large number of components. Second, sometimes the width and height information is not yet available when you need to arrange some control, because the native toolkit components haven‘t been realized.
Each Container object has a layout manager associated with it. A layout manager is an instance of any class that implements the LayoutManager interface. The layout manager is set by the setLayout( ) method. If no call to setLayout( ) is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it.

5. Explain the use of JTextField and any methods associate with it.
Ans.
JTextField is the simplest Swing text component. It is also probably its most widely used text component. JTextField allows you to edit one line of text. It is derived from JTextComponent, which provides the basic functionality common to Swing text components. JTextField uses the Document interface for its model.
Three of JTextField‘s constructors are shown here:
JTextField(int cols) JTextField(String str, int cols)
JTextField(String str)
Here, str is the string to be initially presented, and cols is the number of columns in the text field. If no string is specified, the text field is initially empty. If the number of columns is not specified, the text field is sized to fit the specified string. JTextField generates events in response to user interaction. For example, an ActionEvent is fired when the user presses ENTER. ACaretEvent is fired each time the caret (i.e., the cursor) changes position. (CaretEvent is packaged in javax.swing.event.) Other events are also possible. In many cases, your program will not need to handle these events. Instead, you will simply obtain the string currently in the text field when it is needed. To obtain the text currently in the text field, call getText( ).

6. Explain the use of JList and any methods associated with it.
Ans.
In Swing, the basic list class is called JList. It supports the selection of one or more items from a list. Although the list often consists of strings, it is possible to create a list of just about any object that can be displayed. JList is so widely used in Java that it is highly

unlikely that you have not seen one before. JList provides several constructors. The one used here is
JList(Object[ ] items)
This creates a JList that contains the items in the array specified by items. JList is based on two models. The first is ListModel. This interface defines how access to the list data is achieved. The second model is the ListSelectionModel interface, which defines methods that determine what list item or items are selected. Although a JList will work properly by itself, most of the time you will wrap a JList inside a JScrollPane. This way, long lists will automatically be scrollable, which simplifies GUI design. It also makes it easy to change the number of entries in a list without having to change the size of the JList component.

ListSelectionListener. This listener specifies only one method, called valueChanged( ), which is shown here:
void valueChanged(ListSelectionEvent le)
Here, le is a reference to the object that generated the event. Although ListSelectionEvent does provide some methods of its own, normally you will interrogate the JList object itself to determine what has occurred. Both ListSelectionEvent and ListSelectionListener are packaged in javax.swing.event.
By default, a JList allows the user to select multiple ranges of items within the list, but you can change this behavior by calling setSelectionMode( ), which is defined by JList. It is shown here:
void setSelectionMode(int mode)
Here, mode specifies the selection mode. It must be one of these values defined by ListSelectionModel:
SINGLE_SELECTION SINGLE_INTERVAL_SELECTION
MULTIPLE_INTERVAL_SELECTION.

7. Explain the purpose of JButton and explain any methods associated with it.
Ans.
The JButton class provides the functionality of a push button. JButton allows an icon, a string, or both to be associated with the push button. Three of its constructors are shown here:
JButton(Icon icon)
JButton(String str) JButton(String str, Icon icon)
Here, str and icon are the string and icon used for the button. When the button is pressed, an ActionEvent is generated. Using the ActionEvent object passed to the actionPerformed( ) method of the registered ActionListener, you can obtain the action command string associated with the button. By default, this is the string displayed inside the button. However, you can set the action command by calling setActionCommand( ) on the button. You can obtain the action command by calling getActionCommand( ) on the event object. It is declared like this:
String getActionCommand( )
The action command identifies the button. Thus, when using two or more buttons within the same application, the action command gives you an easy way to determine which button was pressed.

8. With an example explain how to create a swing applet.
Ans.
The second type of program that commonly uses Swing is the applet. Swing-based applets are similar to AWT-based applets, but with an important difference: ASwing applet extends JApplet rather than Applet. JApplet is derived from Applet. Thus, JApplet includes all of the functionality found in Applet and adds support for Swing. JApplet is a top-level Swing container, which means that it is not derived from JComponent. Because JApplet is a top-level container, it includes the various panes described earlier. This means that all components are added to JApplet‘s content pane in the same way that components are added to JFrame‘s content pane.

Swing applets use the same four lifecycle methods as init( ), start( ), stop( ), and destroy( ). Of course, you need override only those methods that are needed by your applet. Painting is accomplished differently in Swing than it is in the AWT, and a Swing applet will not normally override the paint( ) method. All interaction with components in a Swing applet must take place on the event dispatching thread, as described in the previous section. This threading issue applies to all Swing programs.

9. Explain the use of JCheckBox and any methods associated with it.
Ans.
The JCheckBox class provides the functionality of a check box. Its immediate superclass is JToggleButton, which provides support for two-state buttons, as just described. JCheckBox defines several constructors. The one used here is JCheckBox(String str) It creates a check box that has the text specified by str as a label. Other constructors let you specify the initial selection state of the button and specify an icon. When the user selects or deselects a check box, an ItemEvent is generated. You can obtain a reference to the JCheckBox that generated the event by calling getItem( ) on the ItemEvent passed to the itemStateChanged( ) method defined by ItemListener. The easiest way to determine the selected state of a check box is to call isSelected( ) on the JCheckBox instance. In addition to supporting the normal check box operation, JCheckBox lets you specify the icons that indicate when a check box is selected, cleared, and rolled-over. We won‘t be using this capability here, but it is available for use in your own programs.

10. Give one example for swing program

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class classname implements ActionListener
{
  Declare components;

 classname()
 {
  JFrame jfrm=new JFrame("Frame name");
  jfrm.setLayout(null);
  jfrm.setSize(300,300);
  jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  create components;
  add components to frame;

}

 public void actionPerformed(ActionEvent d)
```

```
 {
   Logic required
  }

   public static void main(String args[])
   {
    new classname();

   }
}
```

UNIT IV

Multiple Choice Questions

Questions based on Understanding

1. In a JDBC program, a query is compiled when it is _____
   A.Executed
   B.Initialized
   C.Prepared
   D.Invoked
   Answer: Option C
2. In order to transfer data between a database and an application written in the Java programming language, the JDBC API provides which of these methods?

A. Methods on the ResultSet class for retrieving SQL SELECT results as Java types.
B. Methods on the PreparedStatement class for sending Java types as SQL statement parameters.
C. Methods on the CallableStatement class for retrieving SQL OUT parameters as Java types.
D. All mentioned above Answer: Option D

3. The Acronym for the term UDA is

A. Unified Data Access
B. Universal Data Access
C. Universal Digital Access
D. Uniform Data Access
Answer: Option B

4. Which method is used to establish the connection with the specified url in a Driver Manager class?

A. public static void registerDriver(Driver driver)
B. public static void deregisterDriver(Driver driver)
C. public static Connection getConnection(String url)
D. public static Connection getConnection(String url,String userName,String password)
Answer: Option C

5. Which driver Network connection is indirect that a JDBC client makes to a middleware process that acts as a bridge to the DBMS server?

A. JDBC-Net
B. JDBC-ODBC bridge
C. Native API as basis
D. Native protocol as basis Answer: Option A

6. JDBC technology-based drivers generally fit into how many categories?

A. 4
B. 3
C. 2
D. 5
Answer: Option A

7. JDBC stands for _____

A. Java database connectivity
B. Java database concept
C. Java database communications
D. Java Database convenience
Answer: Option A

8. Which class has traditionally been the backbone of the JDBC architecture?

A. the JDBC driver manager
B. the JDBC driver test suite
C. the JDBC-ODBC bridge

D. All mentioned above Answer: Option A

9. How many steps are used to connect any java application using JDBC?

A. 5
B. 4
C. 3
D. 6

Answer: Option A

10. Name the class acts as an interface between user and drivers in JDBC

A. DriverClass
B. DriverManager
C. JDBCDriver
D. JDBCAppManager

Answer : Option B

11. In RMI Architecture which layer intercepts method calls made by the client/redirects these calls to a remote RMI service?

A. Stub & Skeleton Layer
B. Application Layer
C. Remote Reference Layer
D. Transport Layer

Answer: Option A

12. The abbreviation of RMI is

A. Random Method Invocation
B. Remote Memory Interface
C. Remote Method Invocation
D. Random Method Invocation

Answer: Option C

13. RMI Architecture consists of how many layers?

A. 5
B. 3
C. 4
D. 2

Answer: Option C

14. Which program obtains a remote reference to one or more remote objects on a server and then invokes methods on them in an RMI application?

A. Server
B. Client
C. Both A & B
D. Interface

Answer: Option B

15. Which is an object, acts as a gateway for the client side, all the outgoing requests are routed through it and it resides at the client side and represents the remote object?

A. Stub
B. Skeleton
C. Both A & B
D. None of the above

Answer: Option A

Questions based on Skill

1. In order to execute a SQL statement in a JDBC program, we invoke method

A. executeUpdate method
B. executeRel method
C. executeStmt method

D. executeConn method Answer: Option A

2. Which method is used for an SQL statement that is executed frequently?

A. prepareStatement
B. prepareCall
C. createStatement
D. ExecuteQuery

Answer: Option A

3. What is used to execute parameterized query?

A. Statement interface
B. PreparedStatement interface
C. ResultSet interface
D. Connection object

Answer: Option B

4. Which of the following is an open source DBMS product that runs on UNIX, Linux and Windows.

A. MySQL
B. JSP/SQL
C. JDBC/SQL
D. MS ACCESS

Answer: Option A

5. Name the package that is mandatory to work with JDBC

A. Java.io.*
B. Java.jdbc.*
C. Java.sql.*
D. Java.odbc.*

Answer: Option C

6. An RMI Server is responsible for _____

A. Creating an instance of the remote object
B. Exporting the remote object
C. Binding the instance of the remote object to the RMI registry
D. All mentioned above Answer: Option D

7. What are the exceptions which have to be handled in a RMI client program?

A. RemoteException
B. NotBoundException
C. MalFormedURLException
D. All mentioned above Answer: Option D

8. Which package is used for Remote Method Invocation (RMI)?

A. java.lang.rmi
B. java.lang.reflect
C. java.applet
D. java.rmi

Answer: Option D

9. In RMI, the objects are passed by_____.

A. Value
B. Reference
C. Value and Reference
D. Parameters

Answer: Option A

10. RMI facilitates communication between:

A. 2 Java programs
B. 2 JVMs

C. 2 Machines
D. 2 DNS
Answer: Option A

11. _____is an object whose method can be invoked from another JVM
    A. Stub
    B. Skeleton
    C. Remote object
    D. Gateway

Answer: Option C

12. Name the object that acts as a gateway at the client side in RMI programming
    A. Skeleton
    B. Stub
    C. Remote object
    D. Server

Answer: option B

13. Which of the following is a tool used to generate the stub and the skeleton?

A. javac
B. RMI Registry
C. rmic
D. Java

Answer: Option C

14. Which method of the Naming class in.RMI is used to update the RMI registry on the server machine?

A .rebind ()
B. lookup()
C. Both A & B
D. Binding()

Answer: Option A

15. Which method in naming class specifies a name to the remote object?

A. bind(string name)
B. rebind(string name)
C. Both A & B
D. Lookup(String)

Answer: Option A

Short answer Questions

Questions based on Understanding

1. Discuss the concept of JDBC-ODBC bridge driver.

   JDBC-ODBC bridge driver The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

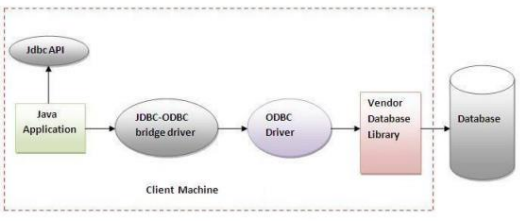


Figure- JDBC-ODBC Bridge Driver

   Advantages: • easy to use. • can be easily connected to any database.
   Disadvantages: • Performance degraded because JDBC method call is converted into the ODBC function calls. • The ODBC driver needs to be installed on the client machine

2. Explain the architecture of JDBC

   Java JDBC is a java API to connect and execute query with the database. JDBC API uses

JDBC drivers to connect with the database. Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language). API (Application programming interface) is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc. JDBC Driver is a software component that enables java application to interact with the database.

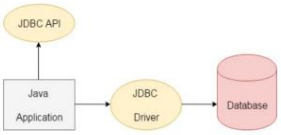


Figure: JDBC Driver

There are 4 types of JDBC drivers:
1) JDBC-ODBC bridge driver:The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC- ODBC bridge driver converts JDBC method calls into the ODBC function calls.
2)Native-API driver(partially java driver):The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.
3)Network Protocol Driver(fully java driver):The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java. 4) Thin Driver(fully java driver):The thin driver converts JDBC calls directly into the vendorspecific database protocol. That is why it is known as thin driver. It is fully written in Java language. Questions based on Application

3. Explain Database connectivity steps using JDBC

There are 5 steps to connect any java application with the database in java using JDBC:
1. 1. **1. Register the driver class:**
The forName() method of Class is used to register the driver class. This method is used to dynamically load the driver class. Syntax of forName() method public static void forName(String className)throws ClassNotFoundException
Example to register the OracleDriver class Class.forName("oracle.jdbc.driver.OracleDriver");

2. Create the connection Object: The getConnection() method of DriverManager class is used to establish connection with the database.
Syntax for getConnection() method
1.public static Connection getConnection(String url)throws SQLException
2.public static Connection getConnection(String url,String name,String password) throws SQLException.
3)Example to establish connection with Oracle database
1.Connection con=DriverManager.getConnection(
2."jdbc:oracle:thin:@localhost:1521:xe","system","password");

3. Create the statement object: The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database. Syntax for createStatement() method public Statement createStatement()throws SQLException Example to create the statement object Statement stmt=con.createStatement();

4. Execute the query: The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get

all the records of a table. Syntax of executeQuery() method public ResultSet executeQuery(String sql)throws SQLException Example to wxwcute query ResultSet rs=stmt.executeQuery("select * from emp"); while(rs.next()){ System.out.println(rs.getInt(1)+" "+rs.getString(2)); }

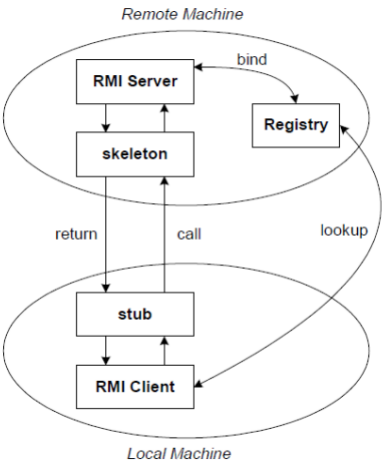
5. Close the connection object: By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method public void close()throws SQLException
con.close();

4. Explain architecture of RMI with the help of a diagram.

Architecture of Remote Method Invocation

The server must first bind its name to the registry. The client lookup the server name in the registry to establish remote references. The Stub serializing the parameters to skeleton, the skeleton invoking the remote method and serializing the result back to the stub. Each remote object has two interfaces.
Client interface – a stub/proxy of the object
Server interface – a skeleton of the object
When a client invokes a remote method, the call is first forwarded to stub. The stub is responsible for sending the remote call over to the server-side skeleton. The stub opening a socket to the remote server, marshaling the object parameters and forwarding the data stream to the skeleton. A skeleton contains a method that receives the remote calls, unmarshals the parameters, and invokes the actual remote object implementation. The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object.



RMI Server The compute engine server accepts tasks from clients, runs the tasks, and returns any results. The server code consists of an interface and a class. The interface defines the methods that can be invoked from the client. Essentially, the interface defines the client's view of the remote object. The class provides the implementation. RMI client. The compute engine is a relatively simple program: it runs tasks that are handed to it. The clients for the compute engine are more complex. A client needs to call the compute engine, but it also has to define the task to be performed by the compute engine

5. Illustrate the concept of stub and skeleton in RMI programming

Stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

- It initiates a connection with remote Virtual Machine (JVM),
- It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
- It waits for the result • It reads (unmarshals) the return value or exception, and
- It finally, returns the value to the caller.

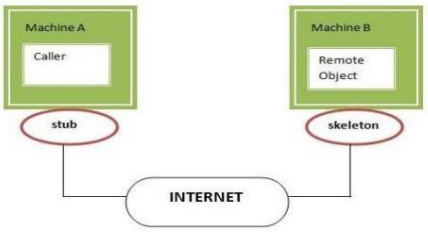


Figure: Stub and Skelton role in RMI

Skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

- It reads the parameter for the remote method
- It invokes the method on the actual remote object, and
- It writes and transmits (marshals) the result to the caller.

Questions based on Application

1. Explain the DriverManager class along with four methods associated with it.

   The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver().

   1) public static void registerDriver(Driver driver): is used to register the given driver with DriverManager.

   2) public static void deregisterDriver(Driver driver): is used to deregister the given driver (drop the driver from the list) with DriverManager.

   3) public static Connection getConnection(String url): is used to establish the connection with the specified url.

   4) public static Connection getConnection(String url,String userName,String password): is used to establish the connection with the specified url, username and password.

2. Define a connection in JDBC. Explain any four methods associated with it.

   A Connection is the session between java application and database. The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData. The Connection interface provide many methods for transaction management like commit(), rollback() etc. Commonly use methods of connection interface

   1) public Statement createStatement(): creates a statement object that can be used to execute SQL queries.

   2) public Statement createStatement(int resultSetType,int resultSetConcurrency): Creates a Statement object that will generate ResultSet objects with the given type and concurrency.

   3) public void setAutoCommit(boolean status): is used to set the commit status.By default it is true.

   4) public void commit(): saves the changes made since the previous commit/rollback permanent.

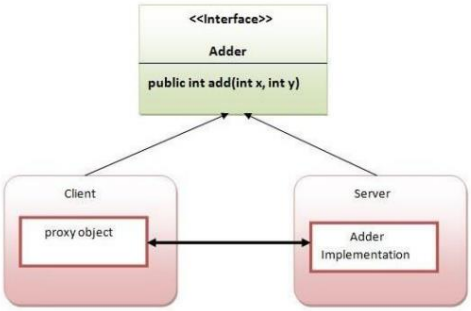
   5) public void rollback(): Drops all changes made since the previous commit/rollback.

   6) public void close(): closes the connection and Releases a JDBC resources immediately.

3. Explain RMI on the Server side and Client side.

   In the client side, RMI requires only two programs. The client application needs only two

files, remote interface and client application. The server also requires two files as interface implementation and server application. In the rmi application, both client and server interact with the remote interface. The client application invokes methods on the proxy object, RMI sends the request to the remote JVM. The return value is sent back to the proxy object and then to the client application.



4. Explain the working of Stub in RMI programming.

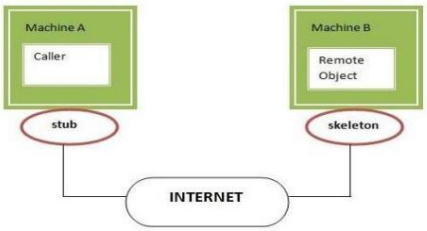


Figure: Stub and Skelton role in RMI

Stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

- It initiates a connection with remote Virtual Machine (JVM),
- It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
- It waits for the result • It reads (unmarshals) the return value or exception, and
- It finally, returns the value to the caller.

5. Explain the working of Skeleton in RMI programming.

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

- It reads the parameter for the remote method
- It invokes the method on the actual remote object, and
- It writes and transmits (marshals) the result to the caller.

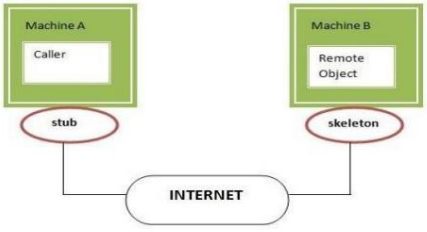


Figure: Stub and Skelton role in RMI

UNIT V

Multiple Choice Questions

Questions based on Application

1. Cookies were originally designed for _____
   a) Client side programming
   b) Server side programming
   c) Both Client side programming and Server side programming
   d) Socket programming
2. The nature of cookies is _____
   a) Volatile
   b) Non-volatile
   c) Transient
   d) Non-transient
3. A cookie can hold _____ amount of data
   a) 1 KB
   b) 2 KB
   c) 3 KB
   d) 4 KB
4. Which of the following are the principal stages in the life cycle of Java Servlet.
   i) Server Initialization
   ii) Servlet Execution
   iii) Servlet Destruction
   iv) Servlet Stop
   A) i, ii, and iii only
   B) i, iii, and iv only
   C) ii, iii, and iv only
   D) All i, ii, iii, and iv only
5. State whether the following statements about the Java servlet life cycle are True or False.
   i) init() and destroy() method will be called only once during the lifetime of the servlet.
   ii) Once the servlet is initialized any request that the servlet container receives will be forwarded to the servlet's execute() method.
   A) i-True, ii-True
   B) i-True, ii-False
   C) i-False, ii-True
   D) i-False, ii-False
6. Which of these ways used to communicate from an applet to servlet?
   a. RMI Communication
   b. HTTP Communication
   c. Socket Communication
   d. All mentioned above
7. Which object of HttpSession can be used to view and manipulate information about a session?
   a. session identifier
   b. creation time
   c. last accessed time
   d. All mentioned above
8. Which cookie it is valid for single session only and it is removed each time when the user closes the browser?

   a. Persistent cookie
   b. **Non-persistent cookie**
   c. All the above
   d. None of the above
9. Which packages represent interfaces and classes for servlet API?
   a. javax.servlet

b. javax.servlet.http
c. Both A & B
d. java.sevlet

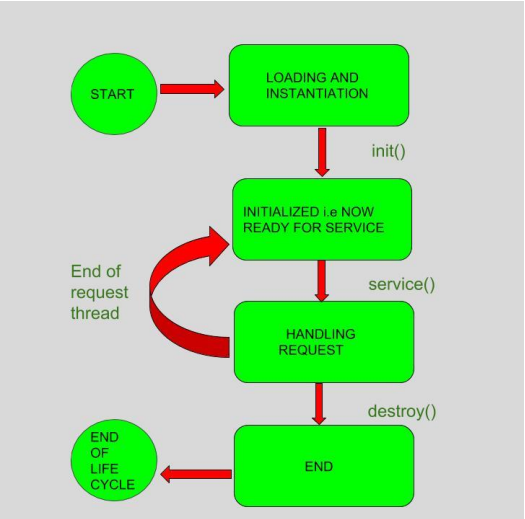
10. What is the lifecycle of a servlet?
    a. Servlet class is loaded
    b. Servlet instance is created
    c. init,Service,destroy method is invoked
    d. All mentioned above
11. Which object is created by the web container at time of deploying the project?
    a. ServletConfig
    b. **ServletContext**
    c. ServletGet
    d. ServletPost
12. Which method in session tracking is used in a bit of information that is sent by a web server to a browser and which can later be read back from that browser?
    a. HttpSession
    b. URL rewriting
    c. Cookies
    d. Hidden form fields
13. How many techniques are used in Session Tracking?
    a. 4
    b. 3
    c. 2
    d. 5
14. Which one of the following scopes does the attribute in servlet is an object that can be set, get or removed?
    a. session scope
    b. request scope
    c. application scope
    d. All mentioned above
15. In HTTP Request what asks for the loopback of the request message, for testing or for troubleshooting?
    a. PUT
    b. OPTIONS
    c. DELETE
    d. TRACE
16. What is javax.servlet.http.HttpServlet?
    A - interface
    B - abstract class
    C - concreate class
    D -Package
17. When service() method of servlet gets called?
    A - The service() method is called when the servlet is first created.
    B - The service() method is called whenever the servlet is invoked.
    C - Both of the above.
    D - The service() method is called when the servlet is destroyed
18. Which of the following code is used to get names of the attributes in servlet?
    A - response.getAttributeNames()
    B - request.getAttributeNames()
    C - Header.getAttributeNames()
    D – Query.getAttributeNames()
19. Which of the following code retrieves the query string that is contained in the request URL after the path?
    A - Header.getQueryString()

B - response.getQueryString()
C - request.getQueryString()
D – getQueryString()

20. **How to create a cookie in servlet?**
    A - Use new operator.
    B - Use request.getCookie() method
    C - Use response.getCookie() method
    D – Use request.setCookie() method
21. **Which of the following code sends a cookie in servlet?**
    A - response.addCookie(cookie);
    B - response.sendCookie(cookie);
    C - response.createCookie(cookie);
    D – request.createCookie(cookie);
22. *HTTP* is a _____protocol
    A. *Stateful*
    B. Stateless
    C. Persistent
    D. Non-persistent
23. Which of the following is/are examples of stateful sessions?
    A. SSH
    B. Telnet
    C. FTP
    D. All the above
24. Which of the following is NOT an example of stateful sessions?
    A. SSH
    B. Telnet
    C. FTP
    D. HTTP
25. Which of the following method in Java changes the name of the cookie?
    A. public void setValue(String value)
    B. public void setName(String value)
    C. public void getValue(String value)
    D. public void getName(String value)

Short Answer Questions

1. Explain the life cycle of a servlet with a neat diagram



The entire life cycle of a Servlet is managed by the **Servlet container** which uses the **javax.servlet.Servlet** interface to understand the Servlet object and manage it.

Stages of the Servlet Life Cycle: The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling.
- Destroying the Servlet.

1. **Loading a Servlet**: The first stage of the Servlet lifecycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages:
    - Initializing the context, on configuring the Servlet with a zero or positive integer value.
    - If the Servlet is not preceding stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request.

    The Servlet container performs two operations in this stage :
    - **Loading:** Loads the Servlet class.
    - **Instantiation:** Creates an instance of the Servlet. To create a new instance of the Servlet, the container uses the no-argument constructor.

2. **Initializing a Servlet**: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the **Servlet.init(ServletConfig)** method which accepts ServletConfig object reference as parameter.

    The Servlet container invokes the **Servlet.init(ServletConfig)** method only once, immediately after the **Servlet.init(ServletConfig)** object is instantiated successfully. This method is used to initialize the resources, such as JDBC datasource.

    Now, if the Servlet fails to initialize, then it informs the Servlet container by throwing the **ServletException** or **UnavailableException**.

3. **Handling request**: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request:
    a. It creates the **ServletRequest** and **ServletResponse** objects. In this case, if this is a HTTP request, then the Web container creates **HttpServletRequest** and **HttpServletResponse** objects which are subtypes of the **ServletRequest** and **ServletResponse** objects respectively.

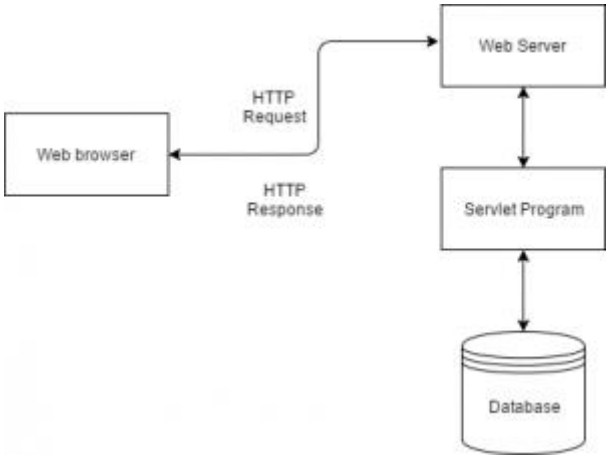
b. After creating the request and response objects it invokes the Servlet.service(ServletRequest, ServletResponse) method by passing the request and response objects.

The **service()** method while processing the request may throw the **ServletException** or **UnavailableException** or **IOException**.

4. **Destroying a Servlet**: When a Servlet container decides to destroy the Servlet, it performs the following operations:
   - It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
   - After currently running threads have completed their jobs, the Servlet container calls the **destroy()** method on the Servlet instance.

2. Explain the servlet architecture with a neat diagram

Servlet Architecture



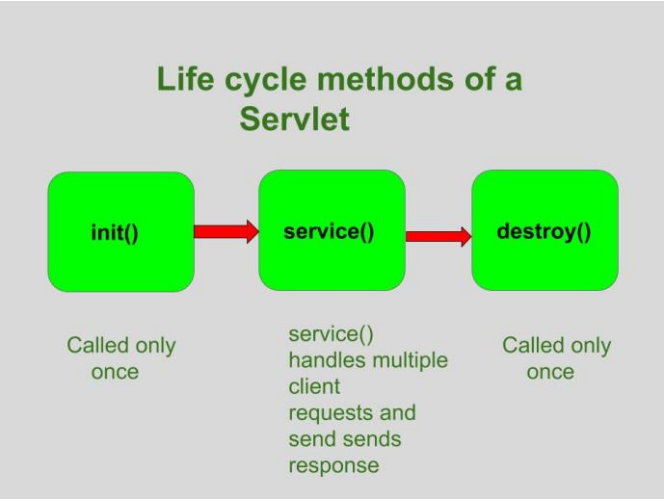
Execution of Servlets basically involves six basic steps:

1. The clients send the request to the webserver.
2. The web server receives the request.
3. The web server passes the request to the corresponding servlet.
4. The servlet processes the request and generates the response in the form of output.
5. The servlet sends the response back to the webserver.
6. The web server sends the response back to the client and the client browser displays it on the screen.

3. Explain the Java methods used during the life cycle of a Servlet

There are three life cycle methods of a Servlet :

- init()
- service()
- destroy()



1. **init() method**: The **Servlet.init()** method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into service.
Syntax:

```
//init() method

public class MyServlet implements Servlet{
  public void init(ServletConfig config) throws ServletException {
      //initialization code
  }
   //rest of code
}
```

2. **service() method**: The **service()** method of the Servlet is invoked to inform the Servlet about the client requests.

- This method uses **ServletRequest** object to collect the data requested by the client.
- This method uses **ServletResponse** object to generate the output content.

Syntax:

```
// service() method
```

3. **destroy() method**: The **destroy()** method runs only once during the lifetime of a Servlet and signals the end of the Servlet instance.
Syntax:

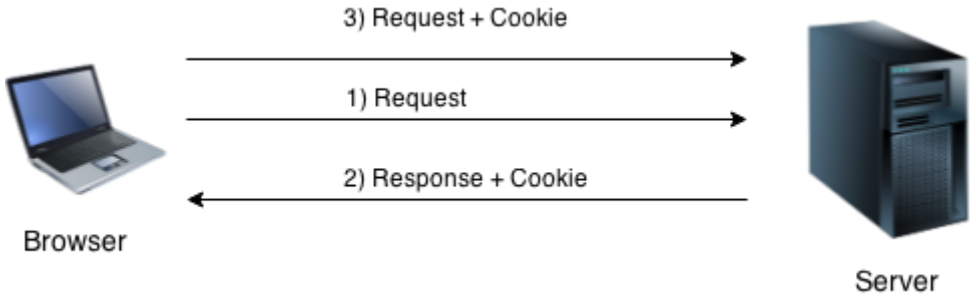
```
//destroy() method

public void destroy()
```

As soon as the **destroy()** method is activated, the Servlet container releases the Servlet instance.

4. Define a cookie. Explain the working of cookies in Java.

A **cookie** is a small piece of information that is persisted between the multiple client requests. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
Web applications are typically a series of Hypertext Transfer Protocol (HTTP) requests and responses. As HTTP is a stateless protocol, information is not automatically saved between HTTP requests. Web applications use cookies to store state information on the client. Cookies can be used to store information about the user, the user's shopping cart, and so on.

Working of Cookies:
By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. Hence, a cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



5. List and explain the different types of cookies available.
   There are 2 types of cookies in servlets.
   1. Non-persistent cookie
   2. Persistent cookie

   Non-persistent cookie (Session Cookies)
   It is **valid for single session** only. It is removed each time when user closes the browser. Session cookies are stored in memory and are accessible as long as the user is using the web application. Session cookies are lost when the user exits the web application. Such cookies are identified by a session ID and are most commonly used to store details of a shopping cart.

   Persistent cookie (Permanent Cookies)
   It is **valid for multiple session**. It is not removed each time when user closes the browser. It is removed only if user logout or signout. Permanent cookies are used to store long-term information such as user preferences and user identification information. Permanent cookies are stored in persistent storage and are not lost when the user exits the application. Permanent cookies are lost when they expire.

6. List any four commonly used methods of the Cookie class in Java

Method	**Description**
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

7. List and describe any four classes available in javax.servlet package

Class Name	Description
GenericServlet	To define a generic and protocol-independent servlet.
ServletContextAttributeEvent	To generate notifications about changes to the attributes of the servlet context of a web application.
ServletContextEvent	To generate notifications about changes to the servlet context of a web application.
ServletInputStream	This class provides an input stream to read binary data from a client request.
ServletOutputStream	This class provides an output stream for sending binary data to the client.
ServletRequestAttributeEvent	To generate notifications about changes to the attributes of the servlet request in an application.
ServletRequestEvent	To indicate lifecycle events for a ServletRequest.
ServletRequestWrapper	This class provides the implementation of the ServletRequest interface that can be subclassed by developers to adapt the request to a Servlet.

ServletResponseWrapper	This class provides the implementation of the ServletResponse interface that can be subclassed by developers to adapt the response from a Servlet.

8. Explain the techniques employed in session tracking in Java.

Session Tracking employs Four Different techniques

1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

1. Cookies

Cookies are little pieces of data delivered by the web server in the response header and kept by the browser. Each web client can be assigned a unique session ID by a web server. Cookies are used to keep the session going. Cookies can be turned off by the client.

2. Hidden Form Field

The information is inserted into the web pages via the hidden form field, which is then transferred to the server. These fields are hidden from the user's view.

3. URL Rewriting

With each request and return, append some more data via URL as request parameters. URL rewriting is a better technique to keep session management and browser operations in sync.

4. HttpSession

A user session is represented by the HttpSession object. A session is established between an HTTP client and an HTTP server using the HttpSession interface. A user session is a collection of data about a user that spans many HTTP requests.

9. Explain the concept of HttpSession interface in Java with a neat diagram

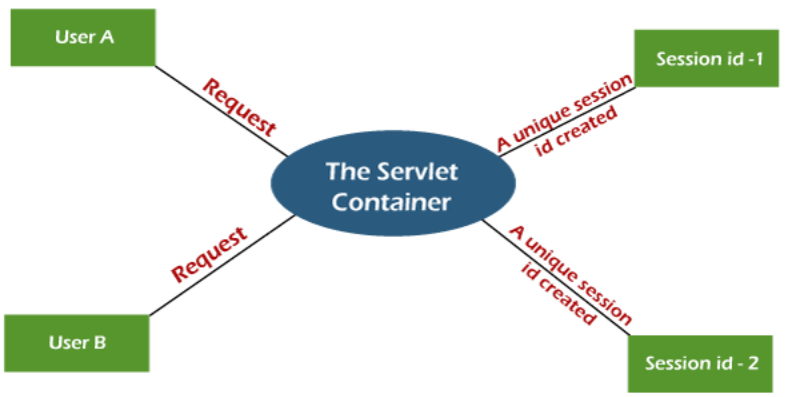
HttpSession Interface

The Java servlets provide the *HttpSession* Interface that gives a way to spot a particular user among multiple page requests or to keep the information about that user. In fact, the Java servlets use the HttpSession interface to establish a connection between the HTTP server and the HTTP client.

The *HttpSession* interface facilitates the servlets to:

- Manipulate and view the information about any session, such as the creation time, the session identifier, and the last accessed time.
- Binding objects to the session, hence; allowing the information about a user to be persistent across the multiple connections.

The following diagram shows the working of the *HttpSession* interface in a session.



User A and User B both are requesting to connect to a server. The servlet container uses the HttpSession interface to connect to the server by creating a unique id for each request. The unique

id is used to identify a user. The unique id can be stored in a request parameter or in a cookie.

10. Explain the different types of sessions with examples

Types of Sessions:

Session comes in two flavours

1. Stateful session
2. Stateless Session

Stateful session

It creates a channel between client and server, session will be closed if any one end gets disturbed. And they need to re-authenticate again to continue their process.
Example: SSH, FTP, Telnet etc

Stateless session

Once the user is authenticated it maintains a token to do the further process, so there is no need to maintain any channel between client and server, the client needs to re-authenticate if the token is lost or expired.
Example: http protocol, all web based application.

@@@@@ ------All the Best----- @@@@@