PART-A unit 1

1. List and explain the java buzzwords **(10Marks) Dec2009**

Sun micro system officially describes java with a list of buzz words or attributes. They are:

- Simple & powerful
- Safe
- Object oriented
- Robust
- Architecture neutral
- Compiled & Interpreted
- Multithreaded
- Easy to learn

The salient features of Java are as follows:

Simple & Powerful: To make the language familiar to the existing programming, java is modeled on C & C++. Java empowers you to express every idea you have in a clean objectoriented way.Safe: Threat of viruses and abuse of resources are every where, java system not only verify
the memory resources but also ensures that no viruses are communicated with the applet. Theabsence of pointers in java ensures that program can not gain access to memory location.Object-oriented: Java is an object-oriented language. Almost every thing in java is object. All the program codes & data reside within object & classes.

Robust: Java is a strictly typed language, because the types must match exactly. It checks yourcode at compile time as well as at run time. Java is a garbage collected language, relieving theprogrammers all memory management problems (i.e., deallocation is completely automatic).

Java incorporates exception handling which captures series of errors and eliminates any riskof crashing the system.

Architecture neutral: Java is the language that is not tied to any particular hardware or operating system. Program developed in java can be executed anywhere on any system. Youcan “write once, run anywhere, anytime forever”. Changes & upgrades in operating system,
processors will not force any changes in java program. It works on Macintosh PC, UNIX &whatever the future platforms can offer.

Interpreted: Java accomplishes architecture neutrality by compiling the java source code intoan intermediate code called “byte codes”, which can be interpreted on any system that has aproper java runtime on it.Multithreaded: Java supports multithreaded programming which allows you to writeprograms that do many things simultaneously.

Easy to learn: The language features feel like the natural way to do things & encourage goodprogramming style. Since object model is both mandatory & simple, you will quicklybecome
acquainted with object oriented style of programming.

BYTE Code is intermediate level code, which is interpreted by the JVM. It is not directly

executable on the machine. This gives java it's "write once and run anywhere" nature. Whenjava program is written and compiled then it will create a .class file which consists of bytecode instructions, understandable to JVM. This class file is system independent. Every system
has it's own JVM.so jvm will convert this byte code into machine language understandable to
that system. So it has run write once and run anywhere nature.
Java achieves architecture neutrality in the following way. Being platform independent wasone of the major objectives for java. Java achieves this independence by introducing anintermediate code representation of compiled java programs. Programs are compiled into abyte code which is then interpreted by platform specific interpreter. The byte code is same forany architecture, IBM compatible, Apple, Sparc, Sun Solaris.
Java programJava compiler The Java Virtual Machine, or JVM, is an abstract computer that runs compiled Java programs.
The JVM is "virtual" because it is generally implemented in software on top of a "real" hardware platform and operating system. All Java programs are compiled for the JVM. Therefore, the JVM must be implemented on a particular platform before compiled Java programs will run on that platform.
A software component needed for developing and executing Java program is called Java Development Kit. JDK is a collection of tools for developing & running java programs. Typically each JDK contains one JRE (Java Runtime Environment, which is an implementation of java virtual machine which actually executes java programs) along withvarious development tools.
Each JDK version adds new APIs and fixes old bugs.
· JDK 1.02 (1995)
· JDK 1.1 (1996)
· Java 2 SDK v 1.2 (also known as JDK 1.2, 1998)
· Java 2 SDK v 1.3 (also known as JDK 1.3, 2000)
· Java 2 SDK v 1.4 (also known as JDK 1.4, 2002)
· JDK 1.5 (also known as Java5, 2004)
· JDK 1.6 (also known as Java6, 2007)
The fundamental mechanisms are known as encapsulation, inheritance and polymorphism.Encapsulation: Encapsulation is a protective wrapper around both the code and data that is being manipulated. It protects the code and data from being arbitrarily accessed by other code.In java basis of encapsulation is a *class*. You create a class that represents an abstraction fora set of objects that share the same structure and behavior. An object is a single instance of a class that retains the structure and behavior as defined by the class. These objects are some
times called *instances of a class.* The individual or data representation of a class is defined bya set of *instance variables*. These variables hold the dynamic state of each instance of a class.The behavior of a class is defined by methods that operate on that instance data. A method is
a message to take some action on an object. Since the goal is to encapsulate complexity, there

are mechanisms for hiding the data declared inside class. Each method or variable in a class may be marked private or public. You can declare private methods and instance data that can
not be accessed by any other code outside the implementation of your class.
Inheritance: Inheritance is the process by which object of one class acquire the properties of objects of another class. Inheritance supports the concept of hierarchical classification. Forexample, the bird robin is a part of the class flying bird, which is again part of a class bird as illustrated in figure below
The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived. In OOP, the concept of inheritance provides the idea of reusability. The new class will have the combined features of both the classes. Java is said to be single inheritance language. Multiple inheritance which is explicitly not a feature of java
Polymorphism: Polymorphism means ability take more than one form or polymorphism means one object, many shapes, a simple concept that allows a method to have multiple implementations that are selected based on the number & type of the arguments passed intothe method invocation. This is known as method overloading. Figure illustrate the method overloadingListed below are some major C++ features that were intentionally omitted from javaJava does not support global variables. it is impossible to create a global variable thatis outside of all the classes.Java has no goto statement
· Java does not use pointers or addresses in memory, because pointers are unsafe. Improper pointer arithmetic causes most of the bugs in today's code.
· There are no header files in java
· Java does not have struct and union construct
· Java does not have preprocessor so we can not use #define, #include statements.
· Java does not support multiple inheritance. Although multiple inheritance is indeed powerful, it is complicated to use correctly and create situations where it's uncertain which method will be executed. For example, if each of the parent classes provide a method X and the derived class does not, it is unclear which X should be invoked.
· Java does not support operator overloading
· Java does not have template classes as in C++. Hence there are no generic functions or classes.
Implementing a java program involves a series of steps. They include:
· Creating the program
· Compiling the program
· Running the program
2. Java has rich set of powerful object classes: Java environment has several key classes for your programs to interact. According to functionality they are grouped into packages. They are:
a. Internet classes include classes for dealing with TCP/IP networking, WWW and HTML and TCP/IP networking. The basic protocols for dealing with the internet are encapsulated in a few simple classes. Extensible implementations of ftp, http, smtp are included which allows us to interact with powerful network services without having to really understand the low-level details of any of these protocols. Since the success of java depends on the WWW and java applets being embedded inside HTML pages. The java classes support these protocols and formats especially well. Java classes are

sent over the network with ease. This is how java applets are loaded over the network
b. Core classes: Several Java core language concepts are implemented as java classes they are:
c. Language support classes: include classes for dealing with primitive types, strings, math functions, multithreading, Exception.
d. Utility classes: include classes for handling vectors, hash tables, stacks, date & time.
e. Input output support classes: classes for managing input & output of data.
f. Abstract graphical user interface classes: java provides a set of classes for implementing GUI in the java release which is called AWT (Abstract Window Toolkit – a library of classes for implementing GUI). Examples: classes for windows, buttons, list, menus etc.

2. Explain different access specifiers is java,with examples. **(06 Marks) Dec2010**

Java provides 3 types of access specifiers.

public: public *keyword* applied to a class, makes it available/visible everywhere. Applied to a
method or variable, completely visible.

Example: public int number;
public void sum( )
{
….
}

private: private fields or methods for a class only visible within that class. Private members
are not visible within subclasses, and are not inherited.

protected: protected members of a class are visible within the class, subclasses and also within
all classes that are in the same package as that class and also to subclasses in other packages.

friendly(default): when no access modifier is specified, the members defaults to friendly level
of access. It makes fields visible within the class, subclasses and also within all classes that are
in the same package as that class.

3. Compare and explain the above two snippets. **(04 Marks) Dec 2009**

i) int num,den;
if(den I= 0 && num |den >2){
}

ii) int num,den;
if(den I=0 & num | den == 0){
}

An object is a single instance of a class that retains the structure and behavior as defined by the class. These objects are some times called *instances of a class*

4 . Explain different access specifiers is java,with examples**. (10 Marks) JAN2009**

Java provides 3 types of access specifiers.

public: public *keyword* applied to a class, makes it available/visible everywhere. Applied to a
method or variable, completely visible.
Example: public int number;
public void sum( )
{
….
}
private: private fields or methods for a class only visible within that class. Private members
are not visible within subclasses, and are not inherited.
protected: protected members of a class are visible within the class, subclasses and also within
all classes that are in the same package as that class and also to subclasses in other packages.
friendly(default): when no access modifier is specified, the members defaults to friendly level
of access. It makes fields visible within the class, subclasses and also within all classes that are
in the same package as that class.
5. Compare and explain the above two snippets. **(10 Marks) Jan2010**
i) int num,den;
if(den I= 0 && num |den >2){
}
ii) int num,den;
if(den I=0 & num | den == 0){
}
An object is a single instance of a class that retains the structure and behavior as defined by the class. These objects are some times called *instances of a class*
6) Write a short notes on Jdk **(06 marks) Dec 2010**

- The Java Development Kit comes with a collection of tools that are used for developing and running java programs They include:
- Javac (java compiler)
- Java (java Interpreter)
- Javap(diassembler)
- Appletviewer (for viewing java applets)
- Javah (for c header files)
- Javadoc (for creating HTML documents)
- Jdb (java debugger).

7) Write a short notes on virtual machine **(04 marks) Dec 2008**

- All language compilers translate source code into machine code for a specific computer
- Java compiler also does the same thing

- Java compiler produces an intermediate code known as byte code for a machine that does not exist. This machine is called the java virtual machine and it exists only inside the computer memory
- The process of compiling a java program into byte code which is also referred to as virtual machine code
- The virtual machine code is not machine specific. The machine specific code (known as machine code) is generated by the java interpreter by acting as an intermediary between the virtual machine and the real machine
- Begin with Java ***source code*** in text files: **Model.java**
- A Java source code compiler produces Java ***byte code***
    - Outputs one file per class: **Model.class**
    - May be standalone or part of an IDE
- A ***Java Virtual Machine*** loads and executes class files
    - May compile them to native code (e.g., x86) internally

8) Difference between literal and whitespace**(06 Marks) May 2009**

Literals

- Literals in java are a sequence of character (digits, letters and other char) that represent constant values to be stored in variables. They are
- Integer literals
- Floating
- Char
- String
- Boolean

White Space

- White space in Java is used to separate the tokens in a Java source file
- White space is required in some places, such as between access modifiers, type names and Identifiers
- Java white space consists of the
- space character **' '** (0x20),
- the tab character (hex 0x09),
- the form feed character (hex 0x0c),
- the line separators characters new line (hex 0x0a) or carriage return (hex 0x0d) characters.

Unit 2

1. Which is the alternative approach to implement multiple inheritances in Java? Explain, with an example. **(06Marks) Dec2010**

An object in java is a block of memory that contains space to store all the instance variables.

Creating an object is also called as instantiating an object. An instance is a individual copy

of the class template with its own set of data called *instance variables*. When you declare a
variable of type class, it has a default value of *null*.
Example: Point p; // here the variable p has null value
Let us see how to create an actual object.
Objects in java are created using *new* operator. The new operator creates an object of specified
class and returns a reference to that object.
Here is an example of creating an object of type 'Point'
Point p = new Point ( );
The method Point( ) is the *default constructor* of the class.
Methods are declared inside of a class definition immediately after the declaration of instance
variables. The *general form* of method declaration is:
type method-name (parameter-list)
{
method-body;
}
Here *type* is any return type this method wishes to return, including *void* in the case where no
return value is desired. In the case where no parameters are desired, the method declaration
should include a pair of empty parentheses.
Example: class Point
{
int x, y;
void init (int a, int b) // method to initialize the variables x & y
{
x=a;
y=b;
}
}
2. Create a try block that is likely to generate three types of exception and incorporate necessary catch blocks to catch and handle them. **(06 Marks) Jan2011**
This is the general form of an exception-handling block:
try
{
//block of code to be monitored for errors
}
catch (ExceptionType1 exOb)
{
//exception handler for ExceptionType1
}
catch (ExceptionType2 exOb)
{
//exception handler for ExceptionType2

```
}
//...
finally
{
//block of code to be executed before try block ends
}
```

DIVIDE-BY-ZERO EXCEPTION

This small program has an expression that causes a divide-by-zero error.

```
class DivideByZero
{
public static void main (String args[])
{
int d = 0;
int a = 42 / d;
}
}
```

The Java run-time system constructs a new exception object when it detects an attempt to divide-by-zero. It then throws this exception. In the above example, there is no exceptional handler to catch the exception. The default handler provided by the Java run-time system will process any exception that is not caught by a program. The output of the above program when executed by the standard Java JDK runtime interpreter:

```
java.lang.ArithemeticException:/ by zero
at DivideByZero.main(DivideByZero.java:4)
```

Although the default exception handler is provided by Java run-time system, you will usually like to handle exception yourself.

The following program includes a try block and a catch clause, which processes the ArithmeticException generated by the division-by-zero:

```
class DivideByZero
{
public static void main (String args[])
{
int d, a;
try //monitor a block of code
{
d = 0;
a = 42 / d;
System.out.println("This will not be printed. ");
}
catch(ArithmeticException e)
{
System.out.println("Division by zero.");
}
```

```
System.out.println("After catch statement.");
}
```

This program generates the following output:
Division by zero.
After catch statement
The call to println() inside the try block is never executed .Once an exception is thrown, program control transfers out of the try block into the catch block. Once the catch statement
has executed, the program continues with the next line in the program following the entire try/
catch mechanism.

3. What are applets? Explain different stages is the cycle of an applet? **(08 Marks) Jan2009**

Applets are used to provide interactive features to web applications that cannot be provided
by HTML. Since Java's bytecode is platform independent, Java applets can be executed by
browsers for many platforms, including Windows, Unix, Mac OS and Linux. There are open
source tools like applet2app which can be used to convert an applet to a stand alone Java application/windows executable/linux executable. This has the advantage of running a Java
applet in offline mode without the need for internet browser software.
Life Cycle of an Applet: Basically, there are four methods in the Applet class on which any
applet is built.

· init: This method is intended for whatever initialization is needed for your applet. It is called after the param attributes of the applet tag.
· start: This method is automatically called after init method. It is also called whenever user returns to the page containing the applet after visiting other pages
· stop: This method is automatically called whenever the user moves away from the page containing applets. You can use this method to stop an animation.
· destroy: This method is only called when the browser shuts down normally.

4. Which is the alternative approach to implement multiple inheritances in Java? Explain, with an example. **(10 Marks) Dec2011**

An object in java is a block of memory that contains space to store all the instance variables.
Creating an object is also called as instantiating an object. An instance is a individual copy
of the class template with its own set of data called *instance variables*. When you declare a
variable of type class, it has a default value of *null*.
Example: Point p; // here the variable p has null value
Let us see how to create an actual object.

Objects in java are created using *new* operator. The new operator creates an object of specified
class and returns a reference to that object.
Here is an example of creating an object of type 'Point'
Point p = new Point ( );
The method Point( ) is the *default constructor* of the class.
Methods are declared inside of a class definition immediately after the declaration of instance
variables. The *general form* of method declaration is:
type method-name (parameter-list)
{
method-body;
}
Here *type* is any return type this method wishes to return, including *void* in the case where no
return value is desired. In the case where no parameters are desired, the method declaration
should include a pair of empty parentheses.
Example: class Point
{
int x, y;
void init (int a, int b) // method to initialize the variables x & y
{
x=a;
y=b;
}
}
5. Create a try block that is likely to generate three types of exception and incorporate necessary catch blocks to catch and handle them. **(06 Marks) dec2010**
This is the general form of an exception-handling block:
try
{
//block of code to be monitored for errors
}
catch (ExceptionType1 exOb)
{
//exception handler for ExceptionType1
}
catch (ExceptionType2 exOb)
{
//exception handler for ExceptionType2
}
//...
finally
{
//block of code to be executed before try block ends

```
}
```

DIVIDE-BY-ZERO EXCEPTION

This small program has an expression that causes a divide-by-zero error.

```
class DivideByZero
{
public static void main (String args[])
{
int d = 0;
int a = 42 / d;
}
}
```

The Java run-time system constructs a new exception object when it detects an attempt to divide-by-zero. It then throws this exception. In the above example, there is no exceptional handler to catch the exception. The default handler provided by the Java run-time system will process any exception that is not caught by a program. The output of the above program when executed by the standard Java JDK runtime interpreter:

```
java.lang.ArithemeticException:/ by zero
at DivideByZero.main(DivideByZero.java:4)
```

Although the default exception handler is provided by Java run-time system, you will usually like to handle exception yourself.

The following program includes a try block and a catch clause, which processes the ArithmeticException generated by the division-by-zero:

```
class DivideByZero
{
public static void main (String args[])
{
int d, a;
try //monitor a block of code
{
d = 0;
a = 42 / d;
System.out.println("This will not be printed. ");
}
catch(ArithmeticException e)
{
System.out.println("Division by zero.");
}
System.out.println("After catch statement.");
}
```

This program generates the following output:

```
Division by zero.
After catch statement
```

The call to println() inside the try block is never executed .Once an exception is thrown, program control transfers out of the try block into the catch block. Once the catch statement
has executed, the program continues with the next line in the program following the entire try/
catch mechanism.

6 What are applets? Explain different stages is the cycle of an applet? **(08 Marks) dec2011**

Applets are used to provide interactive features to web applications that cannot be provided
by HTML. Since Java's bytecode is platform independent, Java applets can be executed by
browsers for many platforms, including Windows, Unix, Mac OS and Linux. There are open
source tools like applet2app which can be used to convert an applet to a stand alone Java application/windows executable/linux executable. This has the advantage of running a Java
applet in offline mode without the need for internet browser software.

Life Cycle of an Applet: Basically, there are four methods in the Applet class on which any
applet is built.

· init: This method is intended for whatever initialization is needed for your applet. It is called after the param attributes of the applet tag.

· start: This method is automatically called after init method. It is also called whenever user returns to the page containing the applet after visiting other pages

· stop: This method is automatically called whenever the user moves away from the page containing applets. You can use this method to stop an animation.

· destroy: This method is only called when the browser shuts down normally.

Unit -3

1. What is the need of synchronization? Explain with an example, how synchronization is implemented in Java? **(10Marks) Dec2009**

One further area of concern within threads is known as "busy waiting." This is the situation
where a thread is conceptually idle, perhaps waiting for some other synchronous processing to
complete, but yet it is still occupying the CPU. To illustrate, consider a spell-checking thread.
It reads a text file and searches for each word in its database. If a word is not found, the thread
composes a list of suggested corrections and notifies the calling process. The calling process
displays a list of the while the thread waits. Eventually the user makes a selection and the

calling process allows the thread to continue. The thread eventually terminates, once the whole
file has been processed.
We might write such code by using a public boolean variable, 'paused,' like so:

```
paused = true;
parent.processWordList (aListOfWords);
// loop until parent process clears 'paused'
while (paused && !terminated) {}
// Continue
```

The thread simply loops continually until the paused variable is set to false. Although this seems intuitive, the ramifications are that the CPU will continue to spend significant time processing the loopingtime that could best be spent servicing other threads. Indeed, any attempt to move windows around on the screen will be noticeably jerky.

```
void sleep (long milliseconds)
void sleep (long milliseconds, int nanoseconds)
```

These methods make the thread pause for the specified number of milliseconds and/or nanoseconds. What is important here is that the thread really does pause and takes no CPU
time. Not all operating systems support time resolutions as small as nanoseconds, and in these
cases the second method simply rounds the number of nanoseconds to the nearest millisecond
and calls the first method.
The sleep methods have been defined in the Java class libraries as being able to throw an InterruptedException. This exception is generated if the sleep method is disturbed in some
way before it completes, such as if System.exit() is called from elsewhere in the application,
shutting the program down. Whether or not you wish to perform any processing to respond
specifically to a sleep being interrupted, Java mandates the exception be caught, hence your
calls to the sleep methods should be wrapped in a try/catch block. This exception was designed
to provide a general mechanism to allow one thread to implement another. Unfortunately this
has not yet been fully implemented.
The thread class provides a method, interrupt(), which sends an interruption to a specified threadpresently this amounts to nothing more than setting a boolean flag. The boolean function
isInterrupted() may be used to query the status of this flag, but unfortunately there is not presently any way to actually interrupt a thread and throw the InterruptedException. So despite
the fact that the exception must be caught it currently isn't useful for anything. Eventually, it

will permit threads to be woken up from their sleep. Because the sleep method is not presently
interruptible, the alternative is to have brief periods of inactivity (sleeping), before querying
the paused status. Our code fragment thus becomes:

```
paused = true;
parent.processWordList (aListOfWords);
// loop until parent process clears 'paused'
while (paused && !terminated)
try { sleep (2000); }
catch (InterruptedException e) {}
// Continue
```

This code tells the thread to sleep for two seconds (2000 milliseconds) in the body of the while
loop. The thread continues to loop but puts much less strain on the CPU. In fact, the thread's
awake time is greatly reduced.

You have to be careful, though, not to make the sleep time too long or the thread will not respond swiftly once the pause flag has been cleared. Because you are using the thread's sleep
method, you have to catch the exception that could be raised (or javac will complain), but you
don't need to specify any code in the body of the exception handler.

2. What is meant by thread priority? How is it assigned? **(10 Marks) Dec2011**

1. A priority tells the operating system how much resource should be given to each thread.

2. A high-priority thread is scheduled to receive more time on the CPU than a low-priority thread. A method called Thread.setPriority() sets the priority of a thread. Thread class constants can be used to set these.

This class demonstrates how to set Priority

```
public class ThreadDemo extends Thread { public void run() {
for (int i = 0; i < 5; i++)
compute();
}
public static void main(String[] args) {
ThreadDemo thread1 = new ThreadDemo();
Thread thread2 = new Thread(new Runnable() {
public void run() {
for (int i = 0; i < 5; i++)
compute();
}
});
// Set the priorities of these two threads, if any are specified
if (args.length >= 1)
thread1.setPriority(Integer.parseInt(args[0]));
if (args.length >= 2)
```

```
thread2.setPriority(Integer.parseInt(args[1]));
// Start the two threads running
thread1.start();
thread2.start();
for (int i = 0; i < 5; i++)
compute();
static ThreadLocal numcalls = new ThreadLocal();
static synchronized void compute() { Integer n = (Integer) numcalls.get();
if (n == null)
n = new Integer(1);
else
n = new Integer(n.intValue() + 1);
numcalls.set(n);
System.out.println(Thread.currentThread().getName() + ": " + n);
for (int i = 0, j = 0; i < 1000000; i++)
j += i;
try { Thread.sleep((int) (Math.random() * 100 + 1));
} catch (InterruptedException e) {
}
Thread.yield();
}
}
```

3. What is the need of synchronization? Explain with an example, how synchronization is implemented in Java? **(10Marks) Jan2009**

One further area of concern within threads is known as "busy waiting." This is the situation where a thread is conceptually idle, perhaps waiting for some other synchronous processing to complete, but yet it is still occupying the CPU. To illustrate, consider a spell-checking thread. It reads a text file and searches for each word in its database. If a word is not found, the thread composes a list of suggested corrections and notifies the calling process. The calling process displays a list of the while the thread waits. Eventually the user makes a selection and the calling process allows the thread to continue. The thread eventually terminates, once the whole file has been processed.

We might write such code by using a public boolean variable, 'paused,' like so:

```
paused = true;
parent.processWordList (aListOfWords);
// loop until parent process clears 'paused'
while (paused && !terminated) {}
// Continue
```

The thread simply loops continually until the paused variable is set to false. Although this seems intuitive, the ramifications are that the CPU will continue to spend significant time

processing the loopingtime that could best be spent servicing other threads. Indeed, any attempt to move windows around on the screen will be noticeably jerky.
void sleep (long milliseconds)
void sleep (long milliseconds, int nanoseconds)
These methods make the thread pause for the specified number of milliseconds and/or nanoseconds. What is important here is that the thread really does pause and takes no CPU
time. Not all operating systems support time resolutions as small as nanoseconds, and in these
cases the second method simply rounds the number of nanoseconds to the nearest millisecond
and calls the first method.
The sleep methods have been defined in the Java class libraries as being able to throw an InterruptedException. This exception is generated if the sleep method is disturbed in some
way before it completes, such as if System.exit() is called from elsewhere in the application,
shutting the program down. Whether or not you wish to perform any processing to respond
specifically to a sleep being interrupted, Java mandates the exception be caught, hence your
calls to the sleep methods should be wrapped in a try/catch block. This exception was designed
to provide a general mechanism to allow one thread to implement another. Unfortunately this
has not yet been fully implemented.
The thread class provides a method, interrupt(), which sends an interruption to a specified threadpresently this amounts to nothing more than setting a boolean flag. The boolean function
isInterrupted() may be used to query the status of this flag, but unfortunately there is not presently any way to actually interrupt a thread and throw the InterruptedException. So despite
the fact that the exception must be caught it currently isn't useful for anything. Eventually, it
will permit threads to be woken up from their sleep. Because the sleep method is not presently
interruptible, the alternative is to have brief periods of inactivity (sleeping), before querying
the paused status. Our code fragment thus becomes:

```
paused = true;
parent.processWordList (aListOfWords);
// loop until parent process clears 'paused'
while (paused && !terminated)
try { sleep (2000); }
catch (InterruptedException e) {}
// Continue
```

This code tells the thread to sleep for two seconds (2000 milliseconds) in the body of the while
loop. The thread continues to loop but puts much less strain on the CPU. In fact, the thread's
awake time is greatly reduced.
You have to be careful, though, not to make the sleep time too long or the thread will not respond swiftly once the pause flag has been cleared. Because you are using the thread's sleep
method, you have to catch the exception that could be raised (or javac will complain), but you
don't need to specify any code in the body of the exception handler.

4. What is meant by thread priority? How is it assigned? **(10 Marks) Dec2009**

1. A priority tells the operating system how much resource should be given to each thread.
2. A high-priority thread is scheduled to receive more time on the CPU than a low-priority thread. A method called Thread.setPriority() sets the priority of a thread. Thread class constants can be used to set these.

This class demonstrates how to set Priority

```
public class ThreadDemo extends Thread { public void run() {
for (int i = 0; i < 5; i++)
compute();
}
public static void main(String[] args) {
ThreadDemo thread1 = new ThreadDemo();
Thread thread2 = new Thread(new Runnable() {
public void run() {
for (int i = 0; i < 5; i++)
compute();
}
});
// Set the priorities of these two threads, if any are specified
if (args.length >= 1)
thread1.setPriority(Integer.parseInt(args[0]));
if (args.length >= 2)
thread2.setPriority(Integer.parseInt(args[1]));
// Start the two threads running
thread1.start();
thread2.start();
for (int i = 0; i < 5; i++)
compute();
static ThreadLocal numcalls = new ThreadLocal();
static synchronized void compute() { Integer n = (Integer) numcalls.get();
if (n == null)
n = new Integer(1);
else
n = new Integer(n.intValue() + 1);
```

```
numcalls.set(n);
System.out.println(Thread.currentThread().getName() + ": " + n);
for (int i = 0, j = 0; i < 1000000; i++)
j += i;
try { Thread.sleep((int) (Math.random() * 100 + 1));
} catch (InterruptedException e) {
}
Thread.yield();
}
}
```

Unit -4

1. Write the steps to create JTable. Write a program to create a table with the column heardings “Fname,Lname,Age” and insert atleast five records in the table and display**.(06 Marks)Dec2010**

Create a JLabel with an image icon

```
import java.awt.FlowLayout;
import java.awt.HeadlessException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class Main extends JFrame {
public Main() throws HeadlessException {
setSize(300, 300);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new FlowLayout(FlowLayout.LEFT));
Icon icon = new ImageIcon("a.png");
JLabel label1 = new JLabel("Full Name :", icon, JLabel.LEFT);
JLabel label2 = new JLabel("Address :", JLabel.LEFT);
label2.setIcon(new ImageIcon("b.png"));
getContentPane().add(label1);
getContentPane().add(label2);
}
public static void main(String[] args) {
new Main().setVisible(true);
}
```

2) Write a short notes Jtoggle button with an example**(10marks) Dec 2011**

2) // Demonstrate JToggleButton.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
  <applet code="jtoggle" width=200 height=80>
  </applet>
*/
public class jtoggle extends JApplet {
  JLabel jlab;
  JToggleButton jtbn;
  public void init() {
    try {
      SwingUtilities.invokeAndWait(
        new Runnable() {
          public void run() {
            makeGUI();
          }
        }
      );
    } catch (Exception exc) {
      System.out.println("Can't create because of " + exc);
    }
  }

  private void makeGUI() {
    // Change to flow layout.
    getContentPane().setLayout(new FlowLayout());
    // Create a label.
    jlab = new JLabel("Button is off.");
    // Make a toggle button.
    jtbn =  new JToggleButton("On/Off");
    // Add an item listener for the toggle button.
    jtbn.addItemListener(new ItemListener() {
      public void itemStateChanged(ItemEvent ie) {
        if(jtbn.isSelected())
          jlab.setText("Button is on.");
        else
          jlab.setText("Button is off.");
      }

    });
    // Add the toggle button and label to the content pane.
     getContentPane().add(jtbn);
     getContentPane().add(jlab);
  }
}
```

3)Write a short notes on Text field with an example **(08 marks) Dec 2008**

```
 // Demonstrate JTextField.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
  <applet code="JTextFieldDemo" width=300 height=50>
  </applet>
*/
public class jtext extends JApplet {
  JTextField jtf;
  public void init() {
    try {
      SwingUtilities.invokeAndWait(
        new Runnable() {
          public void run() {
            makeGUI();
          }
        }
      );
    } catch (Exception exc) {
      System.out.println("Can't create because of " + exc);
    }
  }

  private void makeGUI() {
    // Change to flow layout.
    getContentPane().setLayout(new FlowLayout());
    // Add text field to content pane.
    jtf = new JTextField(15);
    getContentPane().add(jtf);
    jtf.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ae) {
        // Show text when user presses ENTER.
        showStatus(jtf.getText());
      }
    });
  }
}
```

4)Write a short notes on scrollpane **(10 marks) Jan2009**

```
4) //Demonstrate JScrollPane.
import java.awt.*;
import javax.swing.*;
/*
  <applet code="jscroll" width=300 height=250>
```

```
  </applet>
*/
public class jscroll extends JApplet {
  public void init() {
    try {
      SwingUtilities.invokeAndWait(
        new Runnable() {
          public void run() {
            makeGUI();
          }
        }
      );
    } catch (Exception exc) {
      System.out.println("Can't create because of " + exc);
    }
  }

  private void makeGUI() {
    // Add 400 buttons to a panel.
    JPanel jp = new JPanel();
    jp.setLayout(new GridLayout(20, 20));
    int b = 0;
    for(int i = 0; i < 20; i++) {
      for(int j = 0; j < 20; j++) {
        jp.add(new JButton("Button " + b));
        ++b;
      }
    }
    // Create the scroll pane.
    JScrollPane jsp = new JScrollPane(jp);

    // Add the scroll pane to the content pane.
    // Because the default border layout is used,
    // the scroll pane will be added to the center.
    getContentPane().add(jsp, BorderLayout.CENTER);
  }
}
```

5)Write a short notes on List with an example**(10marks) Dec2009**

```
 5) // Demonstrate JList.
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
/*
  <applet code="jlist" width=200 height=120>
  </applet>
```

```
*/
public class jlist extends JApplet {
  JList jlst;
  JLabel jlab;
  JScrollPane jscrlp;
  // Create an array of cities.
  String Cities[] = { "New York", "Chicago", "Houston",
                      "Denver", "Los Angeles", "Seattle",
                      "London", "Paris", "New Delhi",
                      "Hong Kong", "Tokyo", "Sydney" };
  public void init() {
    try {
      SwingUtilities.invokeAndWait(
        new Runnable() {
          public void run() {
            makeGUI();
          }
        }
      );
    } catch (Exception exc) {
      System.out.println("Can't create because of " + exc);
    }
  }
  private void makeGUI() {
    // Change to flow layout.
    getContentPane().setLayout(new FlowLayout());
    // Create a JList.
    jlst = new JList(Cities);
    // Set the list selection mode to single-selection.
    jlst.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

    // Add the list to a scroll pane.
    jscrlp = new JScrollPane(jlst);
    // Set the preferred size of the scroll pane.
    jscrlp.setPreferredSize(new Dimension(120, 90));
    // Make a label that displays the selection.
    jlab = new JLabel("Choose a City");

    // Add selection listener for the list.
    jlst.addListSelectionListener(new ListSelectionListener() {
      public void valueChanged(ListSelectionEvent le) {
        // Get the index of the changed item.
        int idx = jlst.getSelectedIndex();
        // Display selection, if item was selected.
        if(idx != -1)
          jlab.setText("Current selection: " + Cities[idx]);
```

```
      else // Othewise, reprompt.
        jlab.setText("Choose a City");

    }
  });

  // Add the list and label to the content pane.
  getContentPane().add(jscrlp);
  getContentPane().add(jlab);
}
```

Unit 5 PART-B

1 Give and Explain J2EE multifier architecture. **(08 Marks) Dec2009**

```
public class MainClass {
public static void main(String[] args) {
try {
String className = "org.gjt.mm.mysql.Driver";
Class driverObject = Class.forName(className);
System.out.println("driverObject=" + driverObject);
System.out.println("your installation of JDBC Driver OK.");
} catch (Exception e) {
System.out.println("Failed: JDBC Driver Error: " + e.getMessage());
}
}
}
```

2 Describe the various steps of JDBC process, with code snippets **(06 Marks) Jan2011**

1. A JDBC driver allows a Java application/client to communicate with a SQL database.
2. A JDBC driver is a Java class that implements the JDBC's java.sql.Driver interface.
3. A JDBC driver converts program (and typically SQL) requests for a particular database.

Loading a JDBC Driver: Using Class.forName()

```
String className = "org.gjt.mm.mysql.Driver";
Class driverObject = Class.forName(className);
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
public class Main {
public static void main(String[] args) throws Exception {
Connection conn = getHSQLConnection();
Statement st = conn.createStatement();
```

```
st.executeUpdate("create table survey (id int,name varchar(30));");
st.executeUpdate("insert into survey (id,name ) values (1,'nameValue')");
st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM survey");
ResultSetMetaData rsMetaData = rs.getMetaData();
int numberOfColumns = rsMetaData.getColumnCount();
System.out.println("resultSet MetaData column Count=" + numberOfColumns);
rs.close();
st.close();
conn.close();
}
private static Connection getHSQLConnection() throws Exception {
Class.forName("org.hsqldb.jdbcDriver");
String url = "jdbc:hsqldb:mem:data/tutorial";
return DriverManager.getConnection(url, "sa", "");
}
}
```

Using DriverManager.registerDriver()

```
//String className = "org.gjt.mm.mysql.Driver";
try {
// Registers the given driver with the DriverManager.
DriverManager.registerDriver(new org.gjt.mm.mysql.Driver());
// here the class is loaded
}
catch (SQLException e) {
e.printStackTrace();
}
```

To test a JDBC driver installation using Oracle

```
public class MainClass {
public static void main(String[] args) {
try {
String className = "oracle.jdbc.driver.OracleDriver";
Class driverObject = Class.forName(className);
System.out.println("driverObject=" + driverObject);
System.out.println("your installation of JDBC Driver OK.");
}
catch (Exception e) {
System.out.println("Failed: JDBC Driver Error: " + e.getMessage());
}
}
}
```

3. Write a note on database meta interface. **(04 Marks) Dec2011**

```
public class MainClass {
public static void main(String[] args) {
try {
String className = "org.gjt.mm.mysql.Driver";
```

```
Class driverObject = Class.forName(className);
System.out.println("driverObject=" + driverObject);
System.out.println("your installation of JDBC Driver OK.");
} catch (Exception e) {
System.out.println("Failed: JDBC Driver Error: " + e.getMessage());
}
}
}
```

4. Describe the various steps of JDBC process, with code snippets **(06 Marks)Jan2009**

1. A JDBC driver allows a Java application/client to communicate with a SQL database.
2. A JDBC driver is a Java class that implements the JDBC's java.sql.Driver interface.
3. A JDBC driver converts program (and typically SQL) requests for a particular database.

Loading a JDBC Driver: Using Class.forName()

```
String className = "org.gjt.mm.mysql.Driver";
Class driverObject = Class.forName(className);
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
public class Main {
public static void main(String[] args) throws Exception {
Connection conn = getHSQLConnection();
Statement st = conn.createStatement();
st.executeUpdate("create table survey (id int,name varchar(30));");
st.executeUpdate("insert into survey (id,name ) values (1,'nameValue')");
st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM survey");
ResultSetMetaData rsMetaData = rs.getMetaData();
int numberOfColumns = rsMetaData.getColumnCount();
System.out.println("resultSet MetaData column Count=" + numberOfColumns);
rs.close();
st.close();
conn.close();
}
private static Connection getHSQLConnection() throws Exception {
Class.forName("org.hsqldb.jdbcDriver");
String url = "jdbc:hsqldb:mem:data/tutorial";
return DriverManager.getConnection(url, "sa", "");
}
}
```

Using DriverManager.registerDriver()

```
//String className = "org.gjt.mm.mysql.Driver";
try {
// Registers the given driver with the DriverManager.
DriverManager.registerDriver(new org.gjt.mm.mysql.Driver());
// here the class is loaded
}
catch (SQLException e) {
e.printStackTrace();
}
```

To test a JDBC driver installation using Oracle

```
public class MainClass {
public static void main(String[] args) {
try {
String className = "oracle.jdbc.driver.OracleDriver";
Class driverObject = Class.forName(className);
System.out.println("driverObject=" + driverObject);
System.out.println("your installation of JDBC Driver OK.");
}
catch (Exception e) {
System.out.println("Failed: JDBC Driver Error: " + e.getMessage());
}
}
}
```

5. Write a note on database meta interface. (04 Marks) jan2009

```
public class MainClass {
public static void main(String[] args) {
try {
String className = "org.gjt.mm.mysql.Driver";
Class driverObject = Class.forName(className);
System.out.println("driverObject=" + driverObject);
System.out.println("your installation of JDBC Driver OK.");
} catch (Exception e) {
System.out.println("Failed: JDBC Driver Error: " + e.getMessage());
}
}
}
```

Unit 6

1. Explain the lifecycle of a servlet. **(06 Marks) Dec2011**

Servlet Life Cycle
The life cycle of a servlet is controlled by the container in which the servlet has been deployed.
When a request is mapped to a servlet, the container performs the following steps.
1. If an instance of the servlet does not exist, the Web container
a. Loads the servlet class.
b. Creates an instance of the servlet class.
c. Initializes the servlet instance by calling the init method.
Initialization is covered in Initializing a Servlet.
2. Invokes the service method, passing a request and response object.
Service methods are discussed in the section Writing Service Methods.
If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method. Finalization is discussed in Finalizing a Servlet.
2. Describe in detail, how tomcat web server is configured for development of servlet**(06 Marks)Jan2011**
Tomcat
· Tomcat is the Servlet Engine than handles servlet requests for Apache
o Tomcat is a “helper application” for Apache
o It’s best to think of Tomcat as a “servlet container”
· Apache can handle many types of web services
o Apache can be installed without Tomcat
o Tomcat can be installed without Apache
· It’s easier to install Tomcat standalone than as part of Apache
o By itself, Tomcat can handle web pages, servlets, and JSP
· Apache and Tomcat are open source (and therefore free)
3. With a code snippet, explain how session tracking is handled in Java with servlets. **(04 Marks) Dec 2011**

```
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String docType =
"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
"Transitional//EN\">\n";
out.println(docType +
"<HTML>\n" +
"<HEAD><TITLE>Hello</TITLE></HEAD>\n" +
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<H1>Hello World</H1>\n" +
"</BODY></HTML>");
}
}
```

The superclass
· public class HelloServlet extends HttpServlet {

· Every class must extend GenericServlet or a subclass of GenericServlet
o GenericServlet is “protocol independent,” so you could write a servlet to process any protocol
o In practice, you almost always want to respond to an HTTP request, so you extend HttpServlet

4. Explain the lifecycle of a servlet. **(06 Marks)jan2011**
Servlet Life Cycle
The life cycle of a servlet is controlled by the container in which the servlet has been deployed.
When a request is mapped to a servlet, the container performs the following steps.
1. If an instance of the servlet does not exist, the Web container
a. Loads the servlet class.
b. Creates an instance of the servlet class.
c. Initializes the servlet instance by calling the init method.
Initialization is covered in Initializing a Servlet.
2. Invokes the service method, passing a request and response object.
Service methods are discussed in the section Writing Service Methods.
If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method. Finalization is discussed in Finalizing a Servlet.
4. Describe in detail, how tomcat web server is configured for development of servlet(06 Marks)jan2011
Tomcat
· Tomcat is the Servlet Engine than handles servlet requests for Apache
o Tomcat is a “helper application” for Apache
o It’s best to think of Tomcat as a “servlet container”
· Apache can handle many types of web services
o Apache can be installed without Tomcat
o Tomcat can be installed without Apache
· It’s easier to install Tomcat standalone than as part of Apache
o By itself, Tomcat can handle web pages, servlets, and JSP
· Apache and Tomcat are open source (and therefore free)
5. With a code snippet, explain how session tracking is handled in Java with servlets. (**04Marks) May 2009**

```
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String docType =
"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
"Transitional//EN\">\n";
out.println(docType +
"<HTML>\n" +
```

```
"<HEAD><TITLE>Hello</TITLE></HEAD>\n" +
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<H1>Hello World</H1>\n" +
"</BODY></HTML>");
}
}
```

The superclass

· public class HelloServlet extends HttpServlet {

· Every class must extend GenericServlet or a subclass of GenericServlet

o GenericServlet is “protocol independent,” so you could write a servlet to process any protocol

o In practice, you almost always want to respond to an HTTP request, so you extend HttpServlet

Unit 7

1 What is the difference between servlets and JSP? Explain different types of JSP tags **(05Marks) Jan2011**

The main difference between them is, In servlets both the presentation and business logic are place it together. Where as in jsp both are saparated by defining by java beans . In jsp's
the overall code is modulated so the developer who deesn't know about java canwrite jsp pages by simply knowing the additional tages and class names. One more important to be considered is servlet take less time to compile. Jsp is a tool to simplify theprocess and make the
process automate.Both are webapplications used to produce web content that mean dynamic
web pages. Bot are used to take the requests and service the clients.When the JSP engine encounters a tag extension in a JSP attranslation time, it parses the tag librarydescriptor to find
the required tag handler class,and generates code to obtain, and interact with, the taghandler.

The Tag or BodyTag interfaces, one of which must be implemented by any tag handler, For performance reasons, JSP engines will not necessarily instantiate a new tag handler instance every time a tag is encountered in a JSP. Instead, they may maintain a pool of tag
instances, reusing them where possible. When a tag is encountered in a JSP, the JSP engine
will try to find a Tag instance that is not being used, initialize it, use it and release it (but not destroy it), making it available for further use.The programmer has no control over any
pooling that may occur. The repeated use model is similar to a servlet lifecycle, but note one
very important difference: tag handler implementations don't need to concern themselves with

thread safety. The JSP engine will not use an instance of a tag handler to handle a tag unless
it is free. This is good news: as with JSP authoring in general, developers need to worry about
threading issues less often than when developing servlets.
2 Write a JSP to create and read cookies named user id that stores the value JB007.
(06 Marks)Dec2010
The javax.servlet.jsp.tagext.BodyContent Class
The BodyContent class is the key to BodyTag functionality. BodyContent is a subclass of JspWriterthat can be used to manipulate the body content of BodyTag implementationand store it for later retrieval.
The getBodyContent() method of BodyTagSupport returns the BodyContent instance associated
with a particular tag.
To understand the way in which the BodyContent class works, consider how JspWriter objects
are
handled in JSPs using BodyTags: messages 5 and 10 from the sequence diagram above. Before
the BodyTag
begins to evaluate its body content, the generated JSP implementation class includes the following line:
out = pageContext.pushBody();
After the BodyTag's methods have been called, it includes a matching call:
out = pageContext.popBody();
What this means is that each BodyTag has a kind of play area, enabling it to manipulate its
BodyContent
without automatically affecting the JspWriter of the enclosing JSP page or tag. To generate
output, the
BodyTag needs to write the contents of its BodyContent into its enclosing writer explicitly
(see below).
This is the key difference between BodyTags and Tags: Tag implementations have no such
flexibility, and
therefore cannot modify or suppress their body content, although they can prevent it from being evaluated
altogether by returning SKIP_BODY in their implementation of doStartTag().
3. What is RMI? Describe with code snippets RMI at server side. **(10 Marks) Jan2010**
Remote Method Invocation (RMI) facilitates object function calls between Java Virtual Machines (JVMs). JVMs can be located on separate computers - yet one JVM can invoke methods belonging to an object stored in another JVM. Methods can even pass objects that

a foreign virtual machine has never encountered before, allowing dynamic loading of new
classes as required. This is a powerful feature!
RMI Server Example

```
try {
AdderImpl adder = new AdderImpl();
Naming.rebind("adder", adder);
System.out.println("Adder bound");
}
catch (RemoteException re) {
re.printStackTrace();
}
catch (MalformedURLException me) {
me.printStackTrace();


}
```

4. What is the difference between servlets and JSP? Explain different types of JSP tags **(05Marks) Dec 2011**

The main difference between them is, In servlets both the presentation and business logic are place it together. Where as in jsp both are saparated by defining by java beans . In jsp's
the overall code is modulated so the developer who deesn't know about java canwrite jsp pages by simply knowing the additional tages and class names. One more important to be considered is servlet take less time to compile. Jsp is a tool to simplify theprocess and make the
process automate.Both are webapplications used to produce web content that mean dynamic
web pages. Bot are used to take the requests and service the clients.When the JSP engine encounters a tag extension in a JSP attranslation time, it parses the tag librarydescriptor to find
the required tag handler class,and generates code to obtain, and interact with, the taghandler.
The Tag or BodyTag interfaces, one of which must be implemented by any tag handler, For performance reasons, JSP engines will not necessarily instantiate a new tag handler instance every time a tag is encountered in a JSP. Instead, they may maintain a pool of tag
instances, reusing them where possible. When a tag is encountered in a JSP, the JSP engine
will try to find a Tag instance that is not being used, initialize it, use it and release it (but not destroy it), making it available for further use.The programmer has no control over any
pooling that may occur. The repeated use model is similar to a servlet lifecycle, but note one

very important difference: tag handler implementations don't need to concern themselves with
thread safety. The JSP engine will not use an instance of a tag handler to handle a tag unless
it is free. This is good news: as with JSP authoring in general, developers need to worry about
threading issues less often than when developing servlets.

5. Write a JSP to create and read cookies named user id that stores the value JB007. **(06 Marks) Dec2010**

The javax.servlet.jsp.tagext.BodyContent Class

The BodyContent class is the key to BodyTag functionality. BodyContent is a subclass of JspWriterthat can be used to manipulate the body content of BodyTag implementationand store it for later retrieval.

The getBodyContent() method of BodyTagSupport returns the BodyContent instance associated
with a particular tag.

To understand the way in which the BodyContent class works, consider how JspWriter objects
are
handled in JSPs using BodyTags: messages 5 and 10 from the sequence diagram above. Before
the BodyTag
begins to evaluate its body content, the generated JSP implementation class includes the following line:

out = pageContext.pushBody();

After the BodyTag's methods have been called, it includes a matching call:

out = pageContext.popBody();

What this means is that each BodyTag has a kind of play area, enabling it to manipulate its
BodyContent
without automatically affecting the JspWriter of the enclosing JSP page or tag. To generate
output, the
BodyTag needs to write the contents of its BodyContent into its enclosing writer explicitly
(see below).

This is the key difference between BodyTags and Tags: Tag implementations have no such
flexibility, and
therefore cannot modify or suppress their body content, although they can prevent it from being evaluated
altogether by returning SKIP_BODY in their implementation of doStartTag().

6.. What is RMI? Describe with code snippets RMI at server side. (10 Marks)jan2011

Remote Method Invocation (RMI) facilitates object function calls between Java Virtual Machines (JVMs). JVMs can be located on separate computers - yet one JVM can invoke

methods belonging to an object stored in another JVM. Methods can even pass objects that
a foreign virtual machine has never encountered before, allowing dynamic loading of new
classes as required. This is a powerful feature!

RMI Server Example

```
try {
AdderImpl adder = new AdderImpl();
Naming.rebind("adder", adder);
System.out.println("Adder bound");
}
catch (RemoteException re) {
re.printStackTrace();
}
catch (MalformedURLException me) {
me.printStackTrace();
}
```

Unit 8

1. What is deployment descriptor? List the deployment descriptor for EJB 1.1 **(10 Marks)Dec2010**

EJB Deployment Descriptor editor

The EJB deployment descriptor editor is used to modify EJB JAR files and associated Java™ files.The EJB deployment descriptor editor is organized with pages and sections that represent the various properties and settings in the EJB deployment descriptor. Additionally, the editor includes sections and pages related specifically to WebSphere® Application Server bindings and extensions to the EJB specification.The core function is typically located at the top of an editor page. To see core pages and sections, set focus on the editor and press alt-shift-c. The core pages, sections, headers, and tabs will highlight blue and remain in this state until you

press alt-shift-c again. The extensions and bindings are usually nested sections and found at
the bottom of the editor pages. Collapsing a section hides the content, but leaves the heading
information. This is useful in filtering through the data and properties on each page. The editor
remember the sections that you collapse when you close and reopen the editor. Also, you can
resize sections by dragging a hidden border at the end or beginning of each section.
Another important feature of the editor is the enhanced wizard support. You can open wizards
from the editor that walk you through creation and modification of the various elements. The
wizards help you determine problems while stepping through the creation process. If an error
occurs, or if you enter invalid data, the wizard displays a warning or error message at the top
of the wizard page. Wizards offer the ability to create or edit many objects at one time, and you
can work on multiple beans at the same time.The EJB editor modifies the following resources:

· ejb-jar.xml
· ibm-ejb-jar-bnd.xmi
· ibm-ejb-jar-ext.xmi
· ibm-ejb-access-bean.xml
· ws-handler.xmi
· webservicesclient.xml
· ibm-webservicesclient-bnd.xmi
· ibm-webservicesclient-ext.xmi

2 Write a note on JAR file,[JAR]. **(10 Marks)Jan2011**

Java Archive (JAR) files are compressed files into which you can store many files. If you place
the many classes that your application, or applet need in a JAR file, you reduce the size of your
distributabels.

To create a JAR file, go to the directory where your files are located and type:

jar -cf ajarfile.jar *.*

and every file and directory in the current directory will be in a file called ajarfile.jar.

If you like to see what the JAR tility adds to the jar file, type:

jar -cvf ajarfile.jar *.*

The "v" option sends the JAR utility to a verbose mode, printing out information about its activities.

If you want to extract the content of an existing JAR file, type:

jar -xf ajarfile.jar

Again, if you like to see files (or directoies) that the JAR utility extracts from the JAR file, use

the "v" option, as in:
jar -xvf ajarfile.jar
If you like see a table of content of a JAR file, type:
jar -tf ajarfile.jar
Again, you can use the "v" option to see some information about the files in the JAR file,
jar -tvf ajarfile.jar
There are other options with the JAR utility, to see them just type:
jar
If you have a UNIX background, you might have realized the definite similariries beteen JAR
options and the options of UNIX tar utility.

4. What is deployment descriptor? List the deployment descriptor for EJB 1.1 **(10 Marks)Jan2011**

EJB Deployment Descriptor editor
The EJB deployment descriptor editor is used to modify EJB JAR files and associated Java™
files.The EJB deployment descriptor editor is organized with pages and sections that represent
the various properties and settings in the EJB deployment descriptor. Additionally, the editor
includes sections and pages related specifically to WebSphere® Application Server bindings
and extensions to the EJB specification.The core function is typically located at the top of an
editor page. To see core pages and sections, set focus on the editor and press alt-shift-c. The
core pages, sections, headers, and tabs will highlight blue and remain in this state until you
press alt-shift-c again. The extensions and bindings are usually nested sections and found at
the bottom of the editor pages. Collapsing a section hides the content, but leaves the heading
information. This is useful in filtering through the data and properties on each page. The editor
remember the sections that you collapse when you close and reopen the editor. Also, you can
resize sections by dragging a hidden border at the end or beginning of each section.
Another important feature of the editor is the enhanced wizard support. You can open wizards
from the editor that walk you through creation and modification of the various elements. The
wizards help you determine problems while stepping through the creation process. If an error
occurs, or if you enter invalid data, the wizard displays a warning or error message at the top

of the wizard page. Wizards offer the ability to create or edit many objects at one time, and you
can work on multiple beans at the same time.The EJB editor modifies the following resources:
· ejb-jar.xml
· ibm-ejb-jar-bnd.xmi
· ibm-ejb-jar-ext.xmi
· ibm-ejb-access-bean.xml
· ws-handler.xmi
· webservicesclient.xml
· ibm-webservicesclient-bnd.xmi
· ibm-webservicesclient-ext.xmi

5 Write a note on JAR file,[JAR]. **(10 Marks)Jan2010**

Java Archive (JAR) files are compressed files into which you can store many files. If you place
the many classes that your application, or applet need in a JAR file, you reduce the size of your
distributabels.
To create a JAR file, go to the directory where your files are located and type:
jar -cf ajarfile.jar *.*
and every file and directory in the current directory will be in a file called ajarfile.jar.
If you like to see what the JAR tility adds to the jar file, type:
jar -cvf ajarfile.jar *.*
The "v" option sends the JAR utility to a verbose mode, printing out information about its activities.
If you want to extract the content of an existing JAR file, type:
jar -xf ajarfile.jar
Again, if you like to see files (or directoies) that the JAR utility extracts from the JAR file, use
the "v" option, as in:
jar -xvf ajarfile.jar
If you like see a table of content of a JAR file, type:
jar -tf ajarfile.jar
Again, you can use the "v" option to see some information about the files in the JAR file,
jar -tvf ajarfile.jar
There are other options with the JAR utility, to see them just type:
jar
If you have a UNIX background, you might have realized the definite similariries beteen JAR
options and the options of UNIX tar utility.

6. What is Enterprise JavaBeans (**4marks) Dec 2009**

- The Enterprise JavaBeans architecture is a component architecture for the development and deployment of object-oriented distributed enterprise-level applications.
- Applications written using the Enterprise JavaBeans architecture is scalable, transactional and multi-user secure.

- These applications may be written once, and deployed on any server platform that supports the Enterprise JavaBeans specification

7. What is Entity Beans **(6 marks) Jan 2011**

- Model entities in a system
  - represent their data and associated behavior
    - one or many relational database tables
    - an object in an object database
    - an entity in a legacy system
  - Nouns: People, Places or Things
    - Customer, Employee, Student
    - City, Building, Hotel Room
    - Order, Organization, Health Benefit