-Inner Classess

- Assertions

Inner Classes

1.Innerclasses [non-statc inner]

2.NestedClasses [static inner]

3.Local Clsses [class written in insde the method]

4.Annonumous class[class without name]

Local Class

Class defind in a method is known as “LocalClass”

class Outer

{

void m()

{

class Inner

{

}

}

}

class L1

{

public static void main(String[] args)

{

System.out.println("Hello World!");

}

}

When we complie 3 class files are generated

1.L1.class - class fle for main method

2.Outer$Inner.class - inner class which is separted from outer class

3.Outer.class - Outer class

(Q). what type of data members can defind in InnerClasses?

Ans. Except Static data members.

(Q). what type of Outer Class data members can be accessed in InnerClass?

Ans. We can Access

* All class Variables which are out side the methods.
* Only ‘Final’ variables which are present in “method”

(Q). How to access the member of Local Class from the outside of the method?

Ans. 1.if u to access the member of Local Class from the inside of the method

Create the object of Inner class in the method

class Outer

{

void m1()

{

Class Inner

{

Int x =100;

}

Inner ob = new Inner();

s.o.p(Ob.x);

}//method

}//outer

2.if u to access the member of Local Class from the Outside of the method

class Outer

{

Object m1()

{

Class Inner

{

Int x =100;

Void m2()

{

Return new Inner();

}

}

Inner ob = new Inner();

s.o.p(Ob.x);

}//method

}//outer

2.NestedClasses [static inner]

Syntax:

Class outer

{

Static class inner

{

}

}

(Q). what type of data members can defind in NestedClasses?

Ans. Difine any type of [all] data members.

(Q). what type of Outer Class data members can be accessed in NestedClass?

Ans. We can Access

* All static data members Directly
* All Non - static data members accessed through Object.

Class outer

{

Class inner

{

m1(){--}

static m2(){--}

}

Inner ob = new Inner();

Ob.m1()//calling non-static data members [new Inner().m1()]

Inner.m2();//calling static data members

}

Class outside

{

Pvsm()

{

Outer.inner ob = new Outer.inner();

Ob.m1();

Outer.inner.m2();

}

}

(Q).

Ans.

3.InnerClasses [non-static inner]

Syntax:

Class outer

{

class inner

{

}

}

(Q). what type of data members can defind in InnerClasses?

Ans. Difine only non-static data members like local classes

(Q). what type of Outer Class data members can be accessed in NestedClass?

Ans. We can Access

* All the membes of Outer class
* We can accesse data members through Object,or this.

Class outer

{

m1(){--}

Class inner

{

m2(){--}

}

}

Class outside

{

Pvsm()

{

Outer ob1 = new Outer();

Ob.m1();

Outer.inner ob2 = new Outer.inner();

Ob2.m2();

}

}

4.AnnonanousClasses [class without name]

Two types

1.Static Annanomos class

2.non-Static Annanomos class

**1.Static Annanomos class**

Syntax:

Class outer

{

static outer ob = new outer() //annamous class B[say], is extends A🡪 class B extends A

{

}

}

**2.Non- Static Annanomos class**

Syntax:

Class outer

{

inner ob = new inner() //without semiclon

{

}

}

(Q). Writing Annamous class from ‘abstract class’

Ans.

(Q). we can Define anonymous class in method

Ans.

**Assertions**

Assertions are used for **“Debugging”** the Progarmme and identifng errors

To implemt assertion in our program we must use the keyword **“assert”**

**It is used for TRUE/FALSE**

Example:

Assert expression1;

Assert expression1:expression2;

Assert expression1;

Example:

Boolean b = new boolean (args[o])

Assert b;

Sop(“hello”);

*If ‘b’ is True then the statement after the assret will be Excecuted*

(Q). How to Enable Asserts?

Ans. Java-ea 🡪 enabling

(Q).

Ans.

ENUM Constant

🡪Enum ‘s are similar to class

🡪 They are Introduced to “Group the Constants” of similar kinds.

Syntax

enum <enum name>

{

1. constants

2. variables

3. constructors

5. static Block

6. Instance Intilization block

7. Static & Non-Static methods

}

Example

enum family

{

amma, nanna, sis;

}

class Demo

{

public static void main(String[] args)

{

System.out.println(family.amma); //amma

System.out.println(family.nanna); //nanna

System.out.println(family.sis); //sis

}

}

The above code convets as follws

Final class family extends java.lang.Enum

{

Public static final Family amma;

Public static final Family nanna;

Public static final Family sis;

}

RULES

1.We can not Create Object for enums

2.The semicolon [;] after Enum constants are “Optional” ,till if no methods Defind at their

2.We can not extends any class in Enums. Bcoz enums are implicitly extends java.lang.Enum

3.The Data members of “enum” class are static. So we use class name to call them.[Ex. Family.amma]

4.we can call methods with Enum as “arguments”

5.we can Define methods in Enums. But if we define a method their should be a named Constant ends with [;]

6.Enums are can be Defind in “Inner class”, but not “Inside a method”

7.Defining Empty body enums also possible , no CE: [ex : enum AB{}]

8.Defining enum with out Name also possible , no CE: [ex : enum {}]

enum Alpha

{

a,b,c,d,e;

}

class EnumWithMethod

{

void m1(Alpha var)

{

System.out.println(var);

}

public static void main(String[] args)

{

EnumWithMethod ob =new EnumWithMethod ();

ob.m1(Alpha.a);

}

}

(Q). what is “Ordinal”?

Ans. Ordinal is an Integer constant assign to the constats of the Enum. This is Like Index

Ex.

Enum A {a,b,c};

Oridinal of ‘a’ = 0 , Oridinal of ‘b’ = 1 , Oridinal of ‘c’ = 2 ,

This can be retrived using following method.

Public final int ordinal();

(Q). Is semicolon is Optional?

Ans. Enum wines

{

KO,KOH,KBL,KNT //**; Optional**

}

Enum wines

{

KO; //**; Mantdatory**

Void m1()

{

}

}

(Q). Methods in java.lang.Enum

Ans. Public final int ordinal(); 🡪 returns the Position of the Constant

Public static enum[] values(); 🡪it stores the all the Constants in array & points to 1st element.

Generics

Generics are used for Type-Safe Operations.

ArrayList is takes the values of any Data.So, ArrayList is Declared as

ArrayList <Object> 🡪 means ArrayList is takes the values of any Data

To put ArrayList can takes the values of String Data.So, By using “Generics” ArrayList is Declared as

ArrayList <String> 🡪 means ArrayList is takes the only String values

















**In some applications some task need to run periodically, for example a application of report generating checks for new database entry after one day and make reports according to the entries then save all entries in company's permanent record.**

**Task Scheduling in JAVA**

[http://www.roseindia.net/images/previous.gif](http://www.roseindia.net/java/index.shtml)    [http://www.roseindia.net/images/bt_home.gif](http://www.roseindia.net/java/index.shtml)  [http://www.roseindia.net/images/next.gif](http://www.roseindia.net/java/index.shtml)

In some applications some task need to run periodically, for example a application of report generating checks for new database entry after one day and make reports according to the entries then save all entries in company's permanent record.

Java provide facility to schedule tasks as per requirement. This is example given below to schedule a simple task to display a message after 3 seconds.

**Methods and classes used in this example :**

java.util.TimerTask class specifies a task that can be scheduled to run once or after scheduled time. We can define action to be performed by this timer task. by using run() method of java.util.TimeTask. This is protected method.

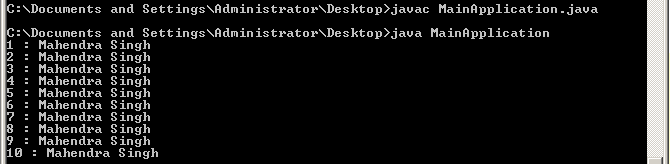
java.util.Timer class provides facility to schedule tasks for future execution in a background thread. In this example we have used schedule() method of this class that specifies task for execution after the specified delay. It takes time delay as a parameter in milliseconds.

**Note :** Use ctrl + C to exit from this program.

**TaskScheduling.java**

import java.util.\*;  
class Task extends TimerTask {  
    
    int count = 1;  
      // run is a abstract method that defines task performed at scheduled time.  
    public void run() {  
        System.out.println(count+" : Mahendra Singh");  
            count++;  
    }  
}  
class TaskScheduling {  
   public static void main(String[] args) {  
       Timer timer = new Timer();  
     
       // Schedule to run after every 3 second(3000 millisecond)  
       timer.schedule( new Task(), 3000);         
   }  
}

**Output of the program :**



import java.util.\*;

class Task extends TimerTask {

int count = 1;

// run is a abstract method that defines task performed at scheduled time.

public void run() {

System.out.println(count+" : Mahendra Singh");

count++;

}

}

class TaskScheduling {

public static void main(String[] args) {

Timer timer = new Timer();

// Schedule to run after every 3 second(3000 millisecond)

timer.schedule( new Task(), 3000);

}

}

**JVM is the main component of Java architecture and it is the part of the JRE (Java Runtime Enviroment) .**

**Java Virtual Machine(JVM)**

[http://www.roseindia.net/images/previous.gif](http://www.roseindia.net/java/index.shtml)    [http://www.roseindia.net/images/bt_home.gif](http://www.roseindia.net/java/index.shtml)  [http://www.roseindia.net/images/next.gif](http://www.roseindia.net/java/index.shtml)

**Ques:-What is Java Virtual Machine?**

**Ans**:- Java Virtual Machine

JVM is the main component of Java architecture and it is the part of the JRE (Java Runtime Enviroment) . It provides the cross platform functionality to java. This is a software process that converts the compiled Java byte code to machine code. Byte code is an intermediary language between Java source and the host system. Most programming language like C and Pascal converts the source code into machine code for one specific type of machine as the machine language vary from system to system . Mostly compiler produce code for a particular system but Java compiler produce code for a virtual machine . JVM provides security to java.

The programs written in Java or the source code translated by Java compiler into byte code and after that the JVM converts the byte code into machine code for the computer one wants to run. JVM is a part of Java Run Time Environment that is required by every operating system requires a different JRE .

The architecture of the JVM is given below . This architecture tell us how the JVM works . Firstly we write the simple java program(source code) the java compiler converts the source code into the bytecode , after that JVM reads this bytecode and converts this into the machine code.

|  |
| --- |
| http://www.roseindia.net/java/images/jvm.gif |

**Ques 2**:- JVM provides portability explain..?

**Ans:-** Implementations of java specification for a variety of CPUs and architectures provides the feature of portability. Foremost, without the availability of a JRE for a given environment, it is impossible to run Java software. JVM forms the part of large system i.e. the Java Runtime Environment (JRE). Each operating system and CPU architecture requires a JRE. JRE consists of a set of base classes i.e. an implementation of the base Java API as well as a JVM. The byte code format is same on all platforms as it runs in the same JVM and it is totally independent from the Operating System and the CPU architecture.JVM is java interpreter as it converts the byte code into machine code for the computer one wants to run.

JRE consists of a number of classes based on JavaAPI and JVM, and without JRE, it is impossible to run Java. So its portability really made it possible in developing write once and run anywhere software .

**How to write First Java program**

**Create a simplest java program :**

|  |
| --- |
| public class MyFirstProgram  {    public static void main(String arr[] )     {      System.out.println(" I am creating my first java program");    }  } |

Save the file with same name as the public class just adding the extension ?**.java**? e.g.  **MyFirstProgram.java.**

Now compile as:

**c:\ javac  MyFirstProgram.java**

This creates a  class file in  with the same name, This is the bytecode form of Java program.

Now to execute this code:  
**c:\ java  MyFirstProgram**

OUTPUT

I am creating my first java program

**Ques 3**: - Memory Management with JVM ?

**Ans**:- Memory Management with JVM :-

the Java language in combination with runtime eliminates the problems of  memory management and corrupted pointers.

1. There is no explicit allocation of memory in Java, memory is allocated only to objects.
2. JVM's heap stores all objects created.
3. JVM ask the operating system for enough memory to run the JVM itself and some free memory for  the  application to create new objects.
4. If the  free memory area is getting too small, the JVM will ask the operating system for more and if there is no more additional memory available from the operating system, then JVM stops the application and issues the "Out of memory error".
5. The Java runtime employs a garbage collector to  reclaim the memory occupied by an object.

**Ques 4**:   Difference amongst JVM Specification, JVM Implementation, JVM Runtime.  
**Ans**:

**I. Java Virtual Machine Specification**

JVM Specification = Java programming language + Java Virtual Machine architecture +  Java class-file format.  **1. Java Programming Language**

**Java Programming Language is a developed by Sun MicroSystems.**  
Programming languages, like natural language follow the syntactic and semantic rules. Java programming language has some form of written specification of syntax and semantics; which are defined only through the sun's official implementation. Java language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode which run on any Java Virtual Machine regardless of any computer architecture.

**2. Java Virtual Machine architecture**

**Java Virtual Machine is responsible for interpreting Java bytecode, and translating it into actions or operating system calls.** JVM  is the main component of Java architecture and it is  the part of the JRE (Java Runtime Enviroment). The Java Virtual Machine provides a platform-independent way of executing code, by abstracting the differences between operating systems and CPU architectures.

**3.  Java class-file format:**

Programs written in Java (source code) are translated by Java compiler into byte code and after that the JVM converts the byte code into machine code for the computer one wants to run. This class file format makes java a portable language.

**II  JVM Implementation:**

**Though implementations of Java Virtual Machines are designed to be compatible, no two JVMs are exactly alike.** For example, garbage collection algorithms vary from one JVM to another, so it becomes impossible to know exactly when memory will be reclaimed. The thread scheduling algorithms are different between one JVM and another , so that it is impossible to accurately predict when one thread will be executed over another.

**We  have multiple implementations of JVM Specifications defined by Sun Microsystems like:**

**(i) Java HotSpot VM:** developed by Sun Microsystems.  
There are two implementations of the Java HotSpot VM:

* **Java HotSpot Client VM**
* **Java HotSpot Server VM**

The main differences of the two implementations are:

* **The Client VM**
  + Reduces application start-up time and memory footprint
* **The Server VM**
  + Maximizes peak operating speed.
  + Intended for running long-running server applications.

**(ii) JRockit JVM :  developed by BEA Systems**

* High performance for server-side applications and optimized for Intel architectures ensuring reliability, scalability, manageability, and flexibility for Java applications.
* Management Console features to connect multiple running JVMs
* Available for both Microsoft Windows 2000 and Red Hat Linux Advanced Server OS.

**III. JVM Runtime:**   
The Java Virtual Machine is responsible for interpreting Java bytecode, and translating further the obtained code into actions or operating system calls. For example, a request to establish a socket connection to a remote machine will involve an operating system call. Different operating systems handle sockets in different ways - but the programmer doesn't need to worry about such details.  It is the responsibility of the JVM to handle these translations, so that the operating system and CPU architecture on which Java software is running is completely irrelevant to the developer.

**JVM Runtime Environment contains the two parts such as:**

**Java API classes:** Java API classes are the predefined classes required for the program compilation and interpretation.

**Java Virtual Machine:** JVM is also a part of the JRE. The Java Virtual Machine is responsible for interpreting Java bytecode, and translating this into actions or operating system calls .

Implementations of java specification for a variety of CPUs and architectures provides the feature of portability. Foremost, without the availability of a JRE for a given environment, it is impossible to run Java software. JVM forms the part of  large system i.e. the Java Runtime Environment (JRE). Each operating system and CPU architecture requires a  JRE. JRE consists of a set of base classes i.e. an implementation of the base Java API as well as a JVM.

\\\\

**Various Commands that are used in java are given below**

**1.javac**

**c:\ javac classname.java**

A Java Compiler javac is a computer program or set of programs which translate java source code into java byte code.

**Usage:** javac <options> <source files>

**where possible options include:**

* g Generate all debugging info
* g:none Generate no debugging in
* g:{lines,vars,source} Generate only some debugging info
* nowarn Generate no warnings
* verbose Output messages about what the compiler is doing
* deprecation Output source locations where deprecated APIs are u

**sed**

* classpath <path> Specify where to find user class files and annotation processors
* cp <path> Specify where to find user class files and annotati

**on processors**

* sourcepath <path> Specify where to find input source files
* bootclasspath <path> Override location of bootstrap class files
* extdirs <dirs> Override location of installed extensions
* endorseddirs <dirs> Override location of endorsed standards path
* proc:{none,only} Control whether annotation processing and/or compilation is done.
* processor <class1>[,<class2>,<class3>...]Names of the annotation processors t

**o run; bypasses default discovery process**

* processorpath <path> Specify where to find annotation processors
* d <directory> Specify where to place generated class files
* s <directory> Specify where to place generated source files
* implicit:{none,class} Specify whether or not to generate class files for

**implicitly referenced files**

* encoding <encoding> Specify character encoding used by source files
* source <release> Provide source compatibility with specified release
* target <release> Generate class files for specific VM version
* version Version information
* help Print a synopsis of standard options
* Akey[=value] Options to pass to annotation processors
* X Print a synopsis of nonstandard options
* J<flag> Pass <flag> directly to the runtime system

**2. java**

**c:\ java classname**

Java interpreter translate the Java bytecode into the code that can be understood by the Operating System.

**Usage:** java [-options] class [args...]

(to execute a class)

or java [-options] -jar jarfile [args...]

(to execute a jar file)

**where options include:**

* client to select the "client" VM
* server to select the "server" VM
* hotspot is a synonym for the "client" VM [deprecated]

**The default VM is client.**

* cp <class search path of directories and zip/jar files>
* classpath <class search path of directories and zip/jar files>

**A ; separated list of directories, JAR archives, and ZIP archives to search for class files.**

* D<name>=<value>

**set a system property**

* verbose[:class|gc|jni]

**enable verbose output**

* version print product version and exit
* version:<value>

**require the specified version to run**

* showversion print product version and continue
* jre-restrict-search | -jre-no-restrict-search

**include/exclude user private JREs in the version search**

* ? -help print this help message
* X print help on non-standard options
* ea[:<packagename>...|:<classname>]
* enableassertions[:<packagename>...|:<classname>]

**enable assertions**

* da[:<packagename>...|:<classname>]
* disableassertions[:<packagename>...|:<classname>]

**disable assertions**

* esa | -enablesystemassertions

**enable system assertions**

* dsa | -disablesystemassertions

**disable system assertions**

* agentlib:<libname>[=<options>]

load native agent library <libname>, e.g. -agentlib:hprof

see also, -agentlib:jdwp=help and -agentlib:hprof=help

* agentpath:<pathname>[=<options>]

**load native agent library by full pathname**

* javaagent:<jarpath>[=<options>]

**load Java programming language agent, see java.lang.instrument**

* splash:<imagepath>

show splash screen with specified image

**3.javap**

**c:\ javap**

This command provides the accessibility of the class members and other features given below .

**Usage:** javap <options> <classes>...

**where options include:**

* c Disassemble the code
* classpath <pathlist> Specify where to find user class files
* extdirs <dirs> Override location of installed extensions
* help Print this usage message
* J<flag> Pass <flag> directly to the runtime system
* l Print line number and local variable tables
* public Show only public classes and members
* protected Show protected/public classes and members
* package Show package/protected/public classes

**and members (default)**

* private Show all classes and members
* s Print internal type signatures
* bootclasspath <pathlist> Override location of class files loaded

**by the bootstrap class loader**

* verbose Print stack size, number of locals and args for methods

If verifying, print reasons for failure

**4. apt (command of bin directory)**

**c:\apt**

This command gives the information about the class path .

**Usage:** apt <apt and javac options> <source files>

**where apt options include:**

* classpath <path> Specify where to find user class files and annotati

**on processor factories**

* cp <path> Specify where to find user class files and annotati

**on processor factories**

* d <path> Specify where to place processor and javac generate

**d class files**

* s <path> Specify where to place processor generated source files
* source <release> Provide source compatibility with specified release
* version Version information
* help Print a synopsis of standard options; use javac -help for more options
* X Print a synopsis of nonstandard options
* J<flag> Pass <flag> directly to the runtime system
* A[key[=value]] Options to pass to annotation processors
* nocompile Do not compile source files to class files
* print Print out textual representation of specified types
* factorypath <path> Specify where to find annotation processor factories
* factory <class> Name of AnnotationProcessorFactory to use; bypasses

**default discovery process**

See javac -help for information on javac options.

**5 .javah**

**c:\javah**

This command include the header files that contains the methods .we need to implement some native methods. You must be wondering about what's native methods.

On the other hand, native methods are the only way to use any system features not provided by the Java Virtual Machine.

**Usage:** javah [options] <classes>

**where [options] include:**

* help Print this help message and exit
* classpath <path> Path from which to load classes
* bootclasspath <path> Path from which to load bootstrap classes
* d <dir> Output directory
* o <file> Output file (only one of -d or -o may be used)
* jni Generate JNI-style header file (default)
* version Print version information
* verbose Enable verbose output
* force Always write output files

<classes> are specified with their fully qualified names (forinstance, java.lang.Object).

**6 . Javadoc**

**c:\javadoc classname.java**

Javadoc is a program that is already included in JDK. We can use Javadoc to run over the source code to produce documentation of our classes in the HTML .

**javadoc:** error - No packages or classes specified.

**usage:** javadoc [options] [packagenames] [sourcefiles] [@files]

* -overview <file> Read overview documentation from HTML file
* public Show only public classes and members
* protected Show protected/public classes and members (default)
* package Show package/protected/public classes and members
* private Show all classes and members
* help Display command line options and exit
* doclet <class> Generate output via alternate doclet
* docletpath <path> Specify where to find doclet class files
* sourcepath <pathlist> Specify where to find source files
* classpath <pathlist> Specify where to find user class files
* exclude <pkglist> Specify a list of packages to exclude
* subpackages <subpkglist> Specify subpackages to recursively load
* breakiterator Compute 1st sentence with BreakIterator
* bootclasspath <pathlist> Override location of class files loaded by the bootstrap class loader
* source <release> Provide source compatibility with specified release
* extdirs <dirlist> Override location of installed extensions
* verbose Output messages about what Javadoc is doing
* locale <name> Locale to be used, e.g. en\_US or en\_US\_WIN
* encoding <name> Source file encoding name
* quiet Do not display status messages
* J<flag> Pass <flag> directly to the runtime system

**Provided by Standard doclet:**

* d <directory> Destination directory for output files
* use Create class and package usage pages
* version Include @version paragraphs
* author Include @author paragraphs
* docfilessubdirs Recursively copy doc-file subdirectories
* splitindex Split index into one file per letter
* windowtitle <text> Browser window title for the documenation
* doctitle <html-code> Include title for the overview page
* header <html-code> Include header text for each page
* footer <html-code> Include footer text for each page
* top <html-code> Include top text for each page
* bottom <html-code> Include bottom text for each page
* link <url> Create links to javadoc output at <url>
* linkoffline <url> <url2> Link to docs at <url> using package list at <url2>
* excludedocfilessubdir <name1>:.. Exclude any doc-files subdirectories with given name.
* group <name> <p1>:<p2>.. Group specified packages together in overview page
* nocomment Supress description and tags, generate only de clarations.
* nodeprecated Do not include @deprecated information
* noqualifier <name1>:<name2>:... Exclude the list of qualifiers from the output.
* nosince Do not include @since information
* notimestamp Do not include hidden time stamp
* nodeprecatedlist Do not generate deprecated list
* notree Do not generate class hierarchy
* noindex Do not generate index
* nohelp Do not generate help link
* nonavbar Do not generate navigation bar
* serialwarn Generate warning about @serial tag
* tag <name>:<locations>:<header> Specify single argument custom tags
* taglet The fully qualified name of Taglet to register
* tagletpath The path to Taglets
* charset <charset> Charset for cross-platform viewing of generated documentation.
* helpfile <file> Include file that help link links to
* linksource Generate source in HTML
* sourcetab <tab length> Specify the number of spaces each tab takes up in the source
* keywords Include HTML meta tags with package, class and member info
* stylesheetfile <path> File to change style of the generated documentation
* docencoding <name> Output encoding name1 error

**7 . jdb**

**c:\jdb -help**

Java debugger helps in finding and the fixing of bugs in Java language programs. The Java debugger is denoted as jdb.

**Usage:** jdb <options> <class> <arguments>

**where options include:**

* help print out this message and exit
* sourcepath <directories separated by ";"> directories in which to look for source files
* attach <address> attach to a running VM at the specified address using standard connector
* listen <address> wait for a running VM to connect at the specified address using standard connector
* listenany wait for a running VM to connect at any available address using standard connector
* launch launch VM immediately instead of waiting for 'run' command
* listconnectors list the connectors available in this VM
* connect <connector-name>:<name1>=<value1>,... connect to target VM using named connector with listed argument values
* dbgtrace [flags] print info for debugging jdb
* tclient run the application in the Hotspot(tm) Performance Engine (Client)
* tserver run the application in the Hotspot(tm) Performance Engine (Server)

**options forwarded to debuggee process:**

* v -verbose[:class|gc|jni] turn on verbose mode
* D<name>=<value> set a system property
* classpath <directories separated by ";"> list directories in which to look for classes
* X<option> non-standard target VM option
* <class> is the name of the class to begin debugging
* <arguments> are the arguments passed to the main() method of <class>

For command help type 'help' at jdb prompt

**8 .Applet viewer**

**C:\ appletviewer classname.html**

Applet viewer is a command line program to run Java applets. It is included in the SDK. It helps you to test an applet before you run it in a browser.

(Q).

Ans.

Naresh java

1.The native modifier indicates that a method is implemented in platform-dependent code, often in C

2.native can be applied only to methods—not classes, not variables, just methods.

3.a native method's body must be a semicolon (;)

4.Var-arg type

5.constructor can't ever, ever, ever, have a return type…ever!

6.Transient : put the varibles cannot participate in Serialization

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Naresh java

Bytecode?

- compiler convers java code to ByteCode

- that ByteCode mst be Excuted by JVM

-So,JVM Must be Installed. Ex.savevid , java plugin

-JDK has both. JDK+JRE+JVM

-main method is Compuory? --> Only for Exceution

- is it possible to call main() implectly?

class A()

{

pvsm()

{

- - - -

}

}

class B()

{

A.main(String[0])

}

--> JVM

--> Acces Modifers

-->Methods , Type of Methods

return

------------------------

-->JVM provides memory only if we use 'Static ' or 'new'

- are Statements are must and Should in methods having return type "void" & "non- void"

Type of Return Statements?

-return; -> sends the CNTL back to calling function WITHOUT Value

-return <value>; -> sends the CNTL back to calling function WITH Value

varible

-----------------------```

Executing method, Excuting varible

Excuting varible - JVM allocates memory when encrted varble

- dff btwn varble callng, method calling

-Local varibles cannot used with out Intialization

1. NON - statc valiables are acceseed fron static block block trough Object only

class A

{

int x = 10;

pvsm

{

S.o.p(x); // wrong CE: --> ob.x; //ri8

}

}

2.Local varibels can be DEcalrd as Stattic? --> no

class A

{

int x =10;//Y

void m1()

{

int b = 21; //Y

static int a = 100; //WRONG. Bcoz static varibles get the memory only at the time of class Loading, but method cannot load automatically.

} IT is Loaded when ever call is come to this method.So, Local varbles cannot Declere as 'Satic'

}

3.the varables craeted in insde the statc block are Static -> what ever the varibels nside the Block they are Local variables only.

4.Transient Varibles

--------------------

If we put 'Transient' keyword to any Varible , that means this varible cannot store in the file in Object Serilzation.

Since, local varibles are not part of Object .So, we dont put 'Transient' to local Varibles

loacl varbles cannot declare as Trancient

class A

{

static transient int a =10; \_/

transient int b =20; \_/

void m1()

{

static transient int c =10; X

transient int d =20; X

}

}

5.Volatile Varibles

---------------------------

If we declared varibles as 'volatile ' then we tell to JVM , we dont want to change the value concurrently by multiple threds

So, Local varibles are not directly accesed by threds .So, we dnt Declre local varibles as 'volatile'

6.Intial & Default values

----------------------------------

Default values applicaple only for class level varibles, not Local varables

7.JVM ARCHITECTURE [insde the jvm by Bill venners]

---------------------------

VM [Virtual mechine] : creation of no.of Exceutable environmnts

how we start JVM PROSESS? --> java filename

Javahotspot Clint vm --> java flname

Javahotspot Server vm --> java -server MyApp

classloadar ->applcation loader, ext loader, bootstrp lod

memarea

heap

pc

stack

native stack

excuton

ninterface

8.Memory used by java Object[138]

----------------------------------------

Shallo size -> only this object

Deep Size -> refferincbg objects

size changes from vm to vm

8 bytes [hashcode] + [reffence]

9.Static members

---------------------------

- can we declare local members as static ? --> no

class A

{

static int a;

int a;//CE: Varibel already Decalred

}

- How compler and JVM search for a Varible Defintion?

1st looks in method block ---not found

goto class area ---not found

CE: cannot find Symbol

class Example2

{

static int a = m1();

public static int m1()

{

System.out.println("a is creadtd");

return 10;

}

static int b = m2();

public static int m2()

{

System.out.println("b is created");

return 20;

}

public static void main(String[] args)

{

System.out.println("main");

System.out.println("a: "+a);

System.out.println("b: "+b);

}

}

output

-----------

a created //varbles are intalzed frst before static blocks

b created

main

a:10

b:20

class Example2

{

static int a = m1();

public static int m1()

{

System.out.println("a is creadtd");

return 10;

}

public static int m2()

{

System.out.println("b is created");

return 20;

}

public static void main(String[] args)

{

System.out.println("main");

System.out.println("a: "+a);

System.out.println("b: "+b);

}

static int b = m2();

}

output

-----------

a created //varible b is last , but how jvm reconizes it?

b created Ans: Exceution of Static [class] memeres are done in two faces

main 1.Identication phase : At the time of Classloding JVM Identifes the Static data membmes.

[Static varibles - allocates the memory , with deft values Static method - Remembers Prototype]

a:10 2.Exceuton phase : varible Exceuton - Assing the values, method excution - Excute logic

b:20

Q. is statc methods are excuted automatcally by jvm like static varables?

A. No, they are excutes only if method is call by the programmer

int a = 10;

a=a //valid but old as same

Q.modularity

A. Divng large code into small

Q.Statc blocks

A. A static bklock isone which has only "Static" keyword

- it is excuted only once when class is loaded for the 1st time

\* "static block always exceutes before main() method"

class StaticBlockDemo

{

static

{

System.out.println("SB");

}

public static void main(String[] args)

{

System.out.println("main");

}

}

output

--------------

sb

main

class StaticBlockDemo

{

static

{

System.out.println("SB");

}

}

comple : ok

excution : sb

exception in thred mainnosuchMethodError : main

Q.nesting of statc blocks are allowed?

A. no

Q. static

{

sop("sb ");

}

static int b = m1()

out

-----

Sb --> static block , static varibles have same priority .so it Depends upon the Order

Q.excute main method at the time of class loading

static

{

main(new String[0])

}

9.NON-STATIC MEMBERS

-----------------------------------------

Q.Object?

A.Object is a continous memory location of a non-static varbles, methods of a class.and points to Object base Address

(or)

it is a encapsulated form of all static & non-static members of a class

Q.what is the outut if we print--> A ob = new A();

s.o.p(ob); classname@12456f A@12456f

Q.which method is used internally Object?

A.toString()

Q. Employee e = null;

s.o.p(e)// null

Q.

int x=10;

int y =20

void m1(int x , int y) --> m1(30,40)

{

x =x; //10 NOT CHANGED SAME AS Before

y = y; //20

}

Q.is RETURN stamet is allowd in contrutor?

A. allowed , but with out value.

return;

return 100; X

10.NON-STATIC BLOCK

----------------------------------------

Non- Static block is Executed , each and Every time when the Object is Created

class A

{

{

S.o.p("iam non static block");

}

}

Q.Non - static block are exceuted before constrctor

Q.what hapapens JVM internally when we craete Object?

A.

- JVM Call the constrctor, but it is not Excute the Body of the Constrctor

- But Instead of it excutes all the 'non -static' varibles, then 'non - static ' blocks

- then finally constror logic will execute

13.Garbage Collection

-------------------------------

Q.Reffrencing Object?

A. A ob = new A();

ob.sum();

Q.Non- Reffrencing Object?

A. new A();

new A().sum();

Q.java.lang.OutOfMemoryError

A. there is no space in Heap

Q.Garbage Collection

A.The proceess of Destroying Unreffereced Objects are Known as "Garbage collection"

Q.Garbage Collection in C++?

A.Manually by calling "destructors"

Q.which thred used by JVM for Garbage Collection proces?

A.'Demon Thred' Internally Runs

Q.How can we Rqst JVM to Do GC?

A. "System.gc()"

"Runtime.getRunTime().gc()"

Q.Garbage Collection Runs in \_\_\_\_memory

A.HEAP

14.Reffrence Variables

------------------------------------

15.OOPs

-----------------------------

Q.Encapsulation?

A.Hiddenig the data of a class

Implemeted by declreing "Private"

setter , getter methods

Q.Final class?

A.the Final class varibles are not FINAL

final class x

{

int a=10; //that means we can access FINAL CLASS DATA whch are not Declared as 'final', but we can modify it

}

class B

{

x ob = new x()

s.o.p(x.a );

}

15.Arrays

--------------------------

16.API

------------------------

> javadoc filename.java

17.Fundamental of Classes

----------------------------------------

Q.Cloning?

A. crete the Dublicate object using curent Object

Q.both have same reffrence?

A.No, Diffrent Refference, but both are in Same state

Q.why 'new '/ 'constructor ' not best for Cloning?

A.if we use them we creates Object with "Intial state", but not current State

class A implemets Clonnable

{

A()

{

- - -

}

public Object clone()

{

A ob = (A)super.clone()

return ob;

}

}

class B

{

pvsm()\

{

}

}

Strings

-------------

String s1 = "a";

String s2 = "a";

SOP(s1==s2)//true

String s1 = new String("a");

String s2 = new String("a");

SOP(s1==s2)//false

this is bcoz of String pooling

Q.Strng pooling?

A.If String are strored with drectly then if both hav same value then they are Refferedby same mem location

Q.what happens if we place stetments after 'Finally block'?

A. C.E : Unreachable Statement, that too when ERROR is come

Other wise it will excute!

class A

{

try {

--

}catch()

{

-- - -

}

finally()

{

- - - -

}

S.o.p ("Unarachable Statement");

}

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.

(Q).

Ans.