Question Bank for M.S.B.T.E Exam for Winter 2015
Chapter 4
Exception Handling & Multithreaded Programming(16 Marks)

Important Questions

1) Define an exception. How it is handled ?

Ans

Defination Exception:

An **Exception** is a condition that is caused by a run-time error in program. When Java interpreter encounters an error such as division by zero, it creates an exception object and throws it to inform that an error has occurred.

Exceptional handling mechanism provides a means to detect errors and throw exceptions, and then to catch exceptions by taking appropriate actions.

Java handles exceptions with 5 keywords:

try,catch,finally,throw,throws

1) try: This block applies a monitor on the statements written inside it. If there exist any exception, the control is transferred to catch or finally block.

2) catch: This block includes the actions to be taken if a particular exception occurs.

3) finally: finally block includes the statements which are to be executed in any case, in case the exception is raised or not.

4)throw: This keyword is generally used in case of user defined exception, to forcefully raise the exception and take the required action.

5)throws: throws keyword can be used along with the method definition to list exceptions which are likely to happen during the execution of that method. In that case,try … catch block is not necessary in the code.

2) What are different types of error ? What is use of throw, throws and finally Statement?

Ans

Errors are broadly classified into two categories:-

1. Compile time errors

2. Runtime errors

Compile time errors: All syntax errors will be detected and displayed by java compiler and therefore these errors are known as **compile time errors**.

The most of common problems are:

- Missing semicolon
- Missing (or mismatch of) bracket in classes & methods
- Misspelling of identifiers & keywords
- Missing double quotes in string
- Use of undeclared variables.
- Bad references to objects.

Runtime errors: Sometimes a program may compile successfully creating the .class file but may not run properly.Such programs may produce wrong results due to wrong logic or may terminate due to errors such as stack overflow. When such errors are encountered java typically generates an error message and aborts the program.

The most common **run-time errors** are:

• Dividing an integer by zero
• Accessing an element that is out of bounds of an array
• Trying to store value into an array of an incompatible class or type
• Passing parameter that is not in a valid range or value for method
• Trying to illegally change status of thread
• Attempting to use a negative size for an array
• Converting invalid string to a number
• Accessing character that is out of bound of a string

throw: If your program needs to throw an exception explicitly, it can be done using **throw** statement. General form of throw statement is:

throw new Throwable subclass;

throw statement is mainly used in case there is user defined exception raised. Throwable instance must be an object of the type Throwable or a subclass of Throwable. The flow of exception stops immediately after the throw statement, any subsequent statements are not executed. The nearest enclosing try block is inspected to see if it has a catch statement that matches the type of exception. If it does find a match, control is transferred to that statement. If not matching catch is found, then the default exception handler halts the program and prints the built in error message.

Example

```
throw new ArithmeticException();
throw new NumberFormatException();
```

throws: If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception.You do this by including a throws clause in the methods declaration. A throws clause lists the types of exception that a method might throw.

General form of method declaration that includes throws clause

```
Type method-name (parameter list) throws exception list
{
// body of method
}
```

Here exception list can be separated by a comma.

finally: It can be used to handle an exception which is not caught by any of the previous catch statements.finally block can be used to handle any statement generated by try block. It may be added immediately after try or after last catch block.

When a finally block is defined, this is guaranteed to execute, regardless of whether or not in exception is thrown. As a result, we can use it to perform certain house-keeping operations such as closing files and releasing system resources.

Syntax

```
try
{……}
catch(..)
{……..}
catch(..)
```

{……..}
finally
{……..}

Or

try
{……..}
finally
{……..}

3) Explain the following cause with respect to exception handling.
a)try b)catch c) finally d)throw

Ans Refer 2nd ans

a) try : Java uses a keyword try to monitor a block of code that is likely to cause an error condition and throw an exception. The try block can have one or more statements that could generate an exception. If any one statement generates an exception, the remaining statements in the block are skipped and execution jumps to the catch block that is placed next to the try block.

b)catch: A catch block defined by the keyword catch "catches" the exception "thrown" by the try block and handles it appropriately. The catch block is added immediately after the try block. The catch block can have one or more statements that are necessary to process the exception. Remember that every try statement should be followed by atleast one catch statement; otherwise compilation error will occur.

Note that the catch statement works like a method definition. The catch statement is passed a single parameter, which is reference to the exception object thrown (by the try block). If the catch parameter matches with the type of exception object, then the exception is caught and statements in the catch block will be executed. Otherwise, the exception is not caught and the default exception handler will cause the execution to terminate.

4) What is Exception ? Give different types of exceptions that could occur during runtime. Why to handle exceptions ?

Ans Defination refer ans 1

Why handle exceptions?

If the exception object is not caught and handled properly, the interpreter displays an error message and will terminate the program. If the execution of the remaining code is to continue, then exception must be caught and handled properly.

List of exceptions

1. ArithmeticException
2. ArrayIndexOutOfBoundsException
3. ArrayStoreException
4. FileNotFoundException
5. IOException
6. NullPointerException
7. NumberFormatException
8. OutOfMemoryException
9. SecurityException

10. StackOverFlowException
11. StringIndexOutOfBoundsException

5) Explain thread priority and method to get and set priority values.

Ans

Definition **Threads** in java are sub programs of main application program and share the same memory space. They are known as light weight threads. A java program requires at least one thread called as main thread. The main thread is actually the main method module which is designed to create and start other threads.

A **Thread** is similar to a program that has a single flow of control.Every thread has a beginning, a body and an end. However, thread is not a program, but runs within a program.

Thread Priority: In java each thread is assigned a priority which affects the order in which it is scheduled for running. Threads of same priority are given equal treatment by the java scheduler.The thread class defines several priority constants as: -

MIN_PRIORITY =1

NORM_PRIORITY = 5

MAX_PRIORITY = 10

Thread priorities can take value from **1-10.**

1) setPriority ()

This method is used to assign new priority to the thread

Syntax: **Thread.setPriority (priority value);**

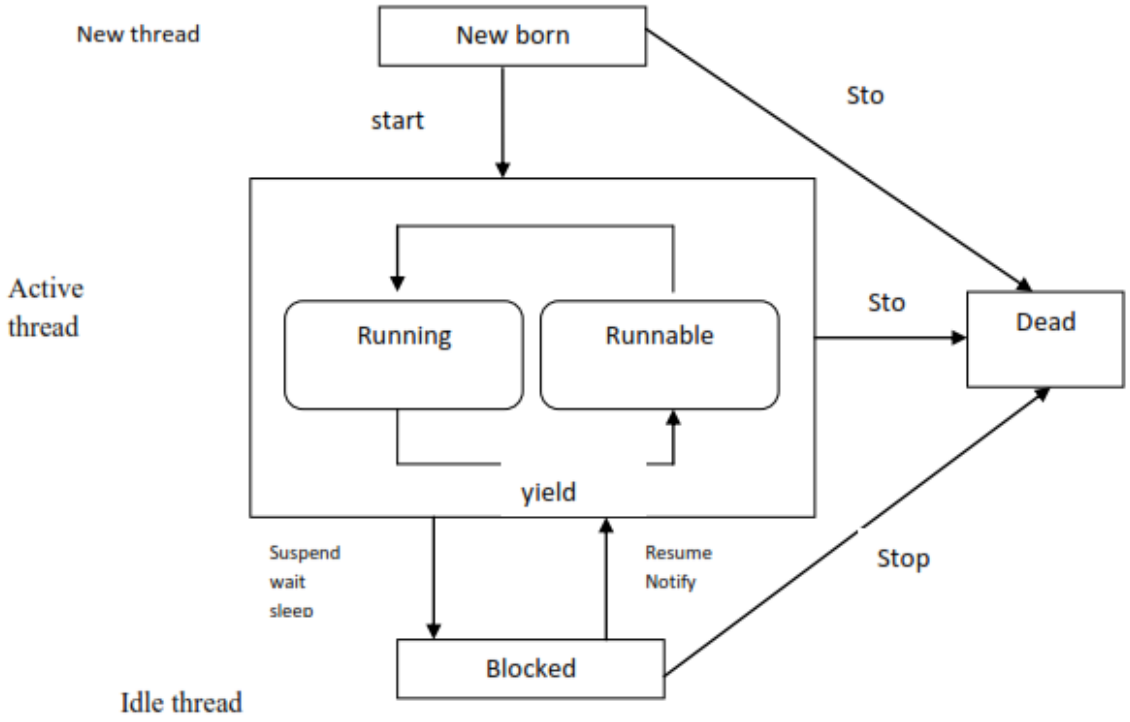
2) getPriority()

It obtain the priority of the thread and returns integer value.

Syntax: int Thread.getPriority ();

6) With suitable diagram explain life cycle of Thread.

Ans



Thread Life Cycle

Thread has **five** different states throughout its life.

1) Newborn State
2) Runnable State
3) Running State
4) Blocked State
5) Dead State

Thread should be in any one state of above and it can be move from one state to another by different methods and ways.

1) Newborn State: When a thread object is created it is said to be in a new born state.When the thread is in a new born state it is not scheduled running from this state it can be scheduled for running by start() or killed by stop(). If put in a queue it moves to runnable state.

2) Runnable State: It means that thread is ready for execution and is waiting for the availability of the processor i.e. the thread has joined the queue and is waiting for execution. If all threads have equal priority then they are given time slots for execution in round robin fashion.The thread that relinquishes control joins the queue at the end and again waits for its turn. A thread can relinquish the control to another before its turn comes by yield().

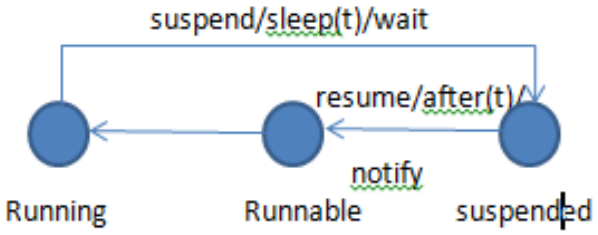
3)Running State: It means that the processor has given its time to the thread for execution. The thread runs until it relinquishes control on its own or it is pre-empted by a higher priority thread.

4)Blocked State: A thread can be temporarily suspended or blocked from entering into the runnable and running state by using either of the following thread method.

suspend() : Thread can be suspended by this method. It can be rescheduled by resume().

wait(): If a thread requires to wait until some event occurs, it can be done using wait method and can be scheduled to run again by notify().

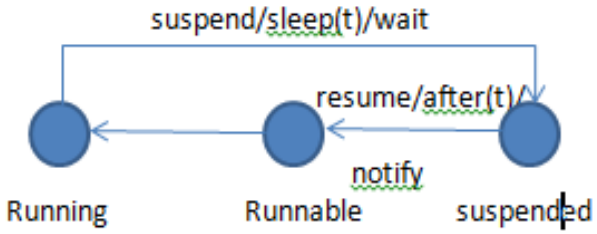
sleep(): We can put a thread to sleep for a specified time period using sleep(time) where time is in ms. It reenters the runnable state as soon as period has elapsed /over.



5)<u>Dead State:</u> Whenever we want to stop a thread form running further we can call its stop(). The statement causes the thread to move to a dead state. A thread will also move to dead state automatically when it reaches to end of the method. The stop method may be used when the premature death is required.

7) Explain following methods related to threads :
a) suspend ( ) b) resume ( ) c) yield ( ) d) wait ( )

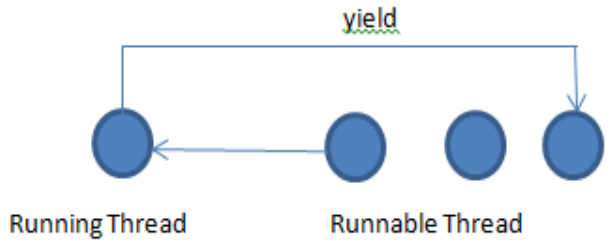
<u>Ans</u>



1) <u>suspend() -</u>
syntax : public void suspend()
This method puts a thread in suspended state i.e blocked and can be resumed using resume() method.

2) <u>resume()</u>
syntax : public void resume()
This method resumes a thread which was suspended using suspend() method.The thread enter in active state i.e Runnable state.

3) yield()
syntax : public static void yield()
The yield() method causes the currently executing thread object to temporarily pause and move to runnable state from running state and allow other threads to execute.



4) wait() and notify()
syntax : public final void wait()

This method causes the current thread to wait until some event occurs and another thread invokes the notify() method or the notifyAll() method for this object.

8) Explain the following methods related to thread
(a) Wait() (b) Notify()

Ans **Refer Ans 7**

wait( ) & notify are required to improve interprocess communication

Winter-12

9) Define thread with example.

Ans **Refer Ans 5 for Defn**

Example:-

```
class Ascending extends Thread
{
public void run()
{
System.out.println(—\n\t Ascending order of numbers::  );
for(int i=1;i<=10 ;i++)
{
System.out.println(—\t  +i);
}
}
}
Class Decending extends Thread
{
Public void run()
{
System.out.println(—\n\t Decending order of numbers::  );
for(int i=10;i>=1;i--)
{
System.out.println(—\t  +i);
}}}

class Threadtest
{
Public static void main(String args[])
{
System.out.println(”One Thread Printing numbers in Ascending order is started” );
new Ascending().start();
System.out.println(“Second Thread Printing numbers in Decending order is started”);
new Decending().start();
}
}
```

Important Programs

Summer-15

10) Write a program to input name and age of a person and throws an user define exception if entered age is negative.

Ans

```
import java.io.*;
class negative extends Exception
{
negative(String msg)
{
super(msg);
}
}
class negativedemo
{
public static void main(String ar[])
{
int age=0;
String name;
InputStreamReader isr=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(isr);
System.out.println("enter age and name of person");
try
{
age=Integer.parseInt(br.readLine());
name=br.readLine();
{
if(age<0)
{
throw new negative("age is negative");
}
else
throw new negative("age is positive");
}}
catch(negative n)
{
System.out.println(n);
}
catch(Exception e)
{}}}
```

Winter-14

11) Write a program to create two threads ; one to print numbers in original order and other to reverse order from 1 to 50.

Ans

```
class original extends Thread
{
public void run()
{
try
{
```

```
for(int i=1; i<=50;i++)
{
System.out.println("\t First Thread="+i);
Thread.sleep(300);
}
}
catch(Exception e)
{}
}
}
class reverse extends Thread
{
public void run()
{
try
{
for(int i=50; i>=1;i--)
{
System.out.println("\t Second Thread="+i);
Thread.sleep(300);
}
}
catch(Exception e)
{}
}
}
class orgrev
{
public static void main(String args[])
{
new original().start();
new reverse().start();
System.out.println("Exit from Main");
}
}
```

Winter-13

12) Write a program to throw a user defined exception "String Mismatch Exception" if two strings are not equal(ignore case).

Ans

```
import java.io.*;
class MyException extends Exception
{
MyException(String msg)
{
super(msg);
}
```

```
}
class ExceptionTest
{
public static void main(String args[])
{
BufferedReader br=new BufferedReader (new InputStreamReader(System.in));
String s1,s2;
try
{
System.out.println("Enter String one and String two ");
s1=br.readLine();
s2=br.readLine();
if(s1.equalsIgnoreCase(s2)) // any similar method which give correct result
{
System.out.println("String Matched");
}
else
{
throw new MyException("String Mismatch Exception");
}
}
catch(MyException e)
{
System.out.println(e);
}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

Summer-13

13) **Write a program to throw a user defined exception as "Invalid Age",if age entered by user is less than eighteen.also mention any two common java exception and their causes**.

Ans

```
import java.lang.Exception;
import java.io.*;
class myException extends Exception
{
myException(String msg)
{
super(msg);
}
}
class agetest
{
```

```
public static void main(String args[])
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
try
{
System.out.println("enter the age : ");
int n=Integer.parseInt(br.readLine());
if(n < 18 )
throw new myException("Invalid Age");
else
System.out.println("Valid age");
}
catch(myException e)
{
System.out.println(e.getMessage());
}
catch(IOException ie)
{}}}
```

Common Java Exceptions

Sr.No	Exception Type	Cause of Exception
1	**ArithmeticException**	Caused by maths errors such as division by zero
2	**ArrayIndexOutOfBoundsException**	Caused by bad array indexes
3	**ArrayStoreException**	Caused when a program tries to store the wrong type of data in an array
4	**FileNotFoundException**	Caused by an attempt access non excesistent file
5	**IOException**	Caused by general I/O failures, such as inability to read from a file
6	**NullPointerException**	Caused by referencing a null object.
7	**NumberFormatException**	Caused when a conversion between strings and number fails
8	**OutOfMemoryException**	Caused when there is not enough memory to allocate a new object.
9	**SecurityException**	Caused when an applet tries to perform an action not allowed by the browsers security settings.
10	**StackOverFlowException**.	Caused when the system runs out of stack space
11	**StringIndexOutOfBoundsException**	Caused when a program attempt to access a nonexistent character position in a string.