Exception Handling.

Page No. 11
Date

(1) Explain different types of Errors in java?

Ans:- There are two types of error's in any programming language

1. Syntax Error / compile time mistake
2. Logical Error / Run time mistake.

- Syntax Error / compile time mistake
- It refers to the mistake done by the programmer with respect to syntax.
- These mistake are identified by the compiler, so we say it as compile Time mistake.

- Logical Error / Runtime mistake
- It refers to the mistake done by the programmer in terms of writing a logic.
- These mistakes are identified by jvm during the execution of a program. So we say it as runtime mistake.

(2) what is an Exception in Java.

Ans:- An unwanted / expected event that disturbs the normal flow of execution of a program is called Exception handling.

- The main objective of Exception handling is to handle the Exception.
- It is available for graceful termination of program.

(3) How you

Ans:- Exceptio

Try :- th
requires

catch :-

Finally :
inte

Ex:-

(4) wh

Ans:- If
ex
Exc
a

(5) what is the difference between exception and error in java.

Ans:- Errors typically happen while an application is running. For instance, out of memory error occurs in case the JVM runsout of memory. on the other hand, exception are mainly caused by the application. For instance, Null pointer Exception happens when an app tries to get through a null object.

(6) Name the different types of exception in java.

Ans:- Based on handling by JVM, there are typically two types of exception in Java.

Checked :- occur during the compilation. Here, the compiler checks whether the exception is handled and throws an error accordingly.

Unchecked :- occur during program execution. These are not detectable during the compilation process.

(7) can we just use try instead of finally and catch blocks? Given an example.

Ans:- No, doing so will show a compilation error. catch or finally block must always accompany try block, we can remove either ~~finally~~ finally block or catch block but never both.

(3)

Ans:-

(2)

Ans:-

```
public static void main(String[] args){
    Daemon Thread t1 = new DaemonThread("t1");
    Daemon Thread t2 = new DaemonThread("t2");
    Daemon Thread t3 = new DaemonThread("t3");
      // setting user thread t1 to Daemon
        t1.setDaemon(true);
    // starting first two thread.
      t1.start();
      t2.start();
    // setting user thread t3 to daemon
        t3.setDaemon(true);
          t3.start();
```

O/P :- t1 is Daemon Thread
t3 is Daemon thread.
T2 is user thread.

(7) what are the wait() and sleep() methods?

Ans: wait():- As the name suggests, it is a non-static method that causes the current threads to wait and go to sleep until some other thread call the notify() or notifyAll() method for the object's monitor (lock). It simply release the lock and mostly used for inter-thread communication. It is define in the object class and should only be called from a synchronized context.

Ex :- synchronized(monitor){
    monitor.wait(); // Here lock is realesed by current Thread.
}

sleep(): As the name suggests, it is a static method that pauses or stops the execution of current thread for some specified periode. It doesn't release the lock while waiting and is mostly used to introduce pause on execution. It is defined in thread class, and no need to call from a synchronized content.

Example:

```
synchronized monitor() {
    Thread.sleep(1000); // lock is held by the current thread.
    // after 1000 milliseconds, the current thread will wake up, or after we call that is interrupt() method.
}
```

(1) Ans:

(2) Ans:

(3) Ans.

(6) How can we create daemon thread.

Ans: We can create daemon threads in java using the thread class setDaemon(true). It is used to mark the current thread as demon thread or user thread. isDaemon() method is generally used to check whether the current thread is daemon or not. If the thread is a daemon, it will return true otherwise it returns false.

Example :- Program to illustrate the use of setDaemon() and isDaemon() method.

```
public class DaemonThread extends Thread.
{
    public DaemonThread(String name).{
        super(name);
}

public void run() {
// checking whether the thread is daemon or not.
if (Thread.currentThread().isDaemon()).
{
sys.o.println(getName() + " is Daemon thread");
} else
{ sys.o.println(getName + " is user thread");
}
}
```

(7)

Ans:

class App{

public static void main (String args[]){
Multithreading Demo obj = new multithreading Demo
obj.start();
}
}

o/p :- My thread is in running state.

(4) Implementing Runnable interface in java.

class Multithreading implements Runnable{
public void run(){
Sysout("My thread is in running state");
}

class App{
psvm(String[] args){
multithreading obj = new multithreading();
Thread t1 = new Thread(obj);
t1.start();
}}

o/p. my thread is in running state.

(05) what is difference between thread and process?

Ans:- Thread: It is refers to the smallest unit of the particular process. It has the ability to execute different parts (referred to as thread) of the program at the same time.

Process:- It simply refers to a program that is in execution. ie an active program. A process can be handled using PCB (process control Block).

⑦ If an exception occurs in a single thread, it will not affect other threads as threads are independent.

⑧ Less resources- intensive than executing multiple processes at the same time.

(3) what is Thread in java?

Ans:- Threads are basically the light weight and smallest unit of processing that can be managed independently by a Schedular. Threads are referred to as part of process that simply let a program execute efficiently with other parts or threads of the process at the same time. using thread one can perform complicated tasks in the easiest way. It is considered the simplest way to take advantage of multiple CPUs available in machine. They share the common address space and are independent to each other.

(4) what are the two ways of implementing threads in java.

Ans: There are basically two ways of implementing threads in java given below

① Extending the thread class.

```
class multithreadingDemo extends Thread {
    public void run() {
    syso("My thread is in running State");
    }
```

(2)

(5)

Ans:-

Multithreading

Page No. 14
Date / / /

① what do you mean by multithreading? why it is important?

Ans: Multithreading means multiple threads and is considered one of the most important features of Java. As the name suggests, it is the ability of a CPU to execute multiple threads independently at the same time but share the process resources simultaneously. Its main purpose is to provide simultaneous execution of multiple thread to utilize the CPU time as much as possible. It is a Java feature where one can subdivide the specific program into two or more threads to make the execution of the program fast and easy.

② what are the benefits of using multithreading?

Ans. There are various benefits of multithreading as given below:-

① Allow program to run continuously even if a part of it is blocked.
② Improve performance as compared to traditional parallel programs that use multiple processes.
③ Allows to write effective programs that utilize maximum CPU time.
④ Improves the responsiveness of complex application or programs.
⑤ Increase use of CPU resources and reduce costs of maintenance.
⑥ Save time and parallelism tasks.

(3) How you can handle Exception in Java? Explain with an example?

Ans:- Exception handling can be performed using:

Try :- the set of statements or code which requires monitoring for an exception is kept under this block

Catch :- This block catches all exceptions that were trapped in the try block.

Finally :- This block is always performed irrespective of the catching of exceptions in try or catch block.

Ex:-
```
class Launch {
    psvm(String[] args) {
        try
        {
            SYSO("Hello world "+" "+1/0);
        }
        catch (ArithmeticException e){
            SYSO("world");
        }
    }
}
```

(4) Why do we need exception handling in Java?

Ans: If there is no try and catch block while an exception occurs, the program will terminate. Exception handling ensures the smooth running of a program without program termination.