Threads(30mins)

Q1.

What will be the result of attempting to compile and run the following program?

public class MyThread implements Runnable

{

String msg = "default";

public MyThread(String s)

{

msg = s;

}

public void run( )

{

System.out.println(msg);

}

public static void main(String args[])

{

new Thread(new MyThread("String1")).run();

new Thread(new MyThread("String2")).run();

System.out.println("end");

}

}

Select 1 correct option.

a The program will compile and print only 'end'.

b It will always print 'String1' 'String2' and 'end', in that order.

c It will always print 'String1' 'String2' in random order followed by 'end'.

d It will always print 'end' first.

e No order is guaranteed.

Q2.

What would be the output of the following code?

class MyRunnable implements Runnable

{

MyRunnable(String name)

{

new Thread(this, name).start();

}

public void run()

{

System.out.println(Thread.currentThread().getName());

}

}

public class TestClass

{

public static void main(String[] args)

{

Thread.currentThread().setName("MainThread");

MyRunnable mr = new MyRunnable("MyRunnable");

mr.run();

}

}

Select 1 correct option.

a MainThread

b MyRunnable

c "MainThread" will be printed twice.

d "MyRunnable" will be printed twice.

e It will print "MainThread" as well as "MyRunnable"

Q3.

The following program will always terminate.

class Base extends Thread

{

static int k = 10;

}

class Incrementor extends Base

{

public void run()

{

for(; k>0; k++)

{

System.out.println("IRunning...");

}

}

}

class Decrementor extends Base

{

public void run()

{

for(; k>0; k--)

{

System.out.println("DRunning...");

}

}

}

public class TestClass

{

public static void main(String args[]) throws Exception

{

Incrementor i = new Incrementor();

Decrementor d = new Decrementor();

i.start();

d.start();

}

}

Select 1 correct option.

a True

b False

Q4.

public class TestClass

{

static StringBuffer sb1 = new StringBuffer();

static StringBuffer sb2 = new StringBuffer();

public static void main(String[] args)

{

new Thread

(

new Runnable()

{

public void run()

{

synchronized(sb1)

{

sb1.append("X");

synchronized(sb2)

{

sb2.append("Y");

}

}

System.out.println(sb1);

}

}

).start();

new Thread

(

new Runnable()

{

public void run()

{

synchronized(sb2)

{

sb1.append("Y");

synchronized(sb1)

{

sb2.append("X");

}

}

System.out.println(sb2);

}

}

).start();

}

}

Select 2 correct options

a It will print 'XX' followed by 'YY'

b It will print 'YY' followed by XX'

c It will print 'XY' followed by 'YX'

d The above code may result in a dead lock and so nothing can be said about the output.

e The output will depend on the threading model of the system JVM is running on.

Q5.

Consider the following program...

public class TestClass implements Runnable

{

int x = 5;

public void run()

{

this.x = 10;

}

public static void main(String[] args)

{

TestClass tc = new TestClass();

new Thread(tc).start(); // 1

System.out.println(tc.x);

}

}

What will it print when run?

Select 1 correct option.

a 5

b 10

c It will not compile.

d Exception at Runtime.

e The output cannot be determined.

Q6.

Consider the following class...

public class Worker implements Runnable

{

Thread w;

public void start()

{

w = new Thread(this);

}

public void run()

{

System.out.println(" Working...");

}

/\* lot of code \*/

}

Assume that the class compiles fine.

What will happen if a Worker object is created and it's start method is called?

Select 1 correct option.

a It will keep on printing "Working...".

b It will not print anything.

c It will print "Working..." just once.

d It will print "Working..." till stop() is called on the Worker object.

e None of the above.

Q7.

Consider the following code:

class MyRunnable implements Runnable

{

public static void main(String[] args)

{

new Thread( new MyRunnable(2) ).start();

}

public void run(int n)

{

for(int i=0; i<n; i++)

{

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

}

}

}

What will be the output when this program is compiled and run from the command line?

Select 1 correct option.

a It'll print "Hello World" once.

b It'll print "Hello World" twice.

c This program will not even compile.

d This will compile but will throw an exception at runtime.

e It will compile and run but it will not print anything.

Q8.

Consider the following code:

class MyClass implements Runnable

{

int n = 0;

public MyClass(int n){ this.n = n; }

public static void main(String[] args)

{

new MyClass(2).run();

new MyClass(1).run();

}

public void run()

{

for(int i=0; i<n; i++)

{

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

}

}

}

What will be the output when this program is compiled and run from the command line?

Select 1 correct option.

a It'll print "Hello World" twice.

b It'll keep printing "Hello World".

c 2 new threads are created by the program.

d 1 new thread is created by the program.

e None of these.

Q9.

Consider the following class:

public class MySecureClass

{

public synchronized void doALotOfStuff(){

try {

LINE1: Thread.sleep(10000);

}catch(Exception e){ }

}

public synchronized void doSmallStuff(){

System.out.println("done");

}

}

Assume that there are two threads. Thread one is executing the doALotOfStuff() method and has just executed LINE 1. Now, Thread two decides to call doSmallStff() method on the same object.

What will happen?

Select 1 correct option.

a done will be printed immediately.

b done will not be printed untill about 10 seconds.

c done will never be printed.

d An IllegalMonitorStateException will be thrown.

e An IllegalThreadStateException will be thrown.

Q10.

What happens when a thread executes wait() method on an object without owning the object's lock?

Select 1 correct option.

a It pauses till somebody releases the lock and then it acquires the lock.

b It causes the object to release all it's locks.

c It acquires the lock and proceeds.

d It forces the other thread that owns the lock to release it.

e It throws an exception.

Q11.

Consider the following method:

public void getLocks(Object a, Object b)

{

synchronized(a)

{

synchronized(b)

{

//do something

}

}

}

and the following instantiations:

Object obj1 = new Object();

Object obj2 = new Object();

obj1 and obj2 are accessible to two different threads and the threads are about to call the getLocks() method.

Assume the first thread calls the method getLocks(obj1, obj2).

Which of the following is true?

Select 1 correct option.

a The second thread should call getLocks(obj2, obj1)

b The second thread should call getLocks(obj1, obj2)

c The second thread should call getLocks() only after first thread exits out of it.

d The second thread may call getLocks() any time and passing parameters in any order.

e None of the above.

Q12.

What will happen if you run the following program...

public class TestClass extends Thread

{

public void run()

{

for(;;);

}

public static void main(String args[])

{

System.out.println("Starting main");

new TestClass().start();

System.out.println("Main returns");

}

}

Select 3 correct options

a It will print "Starting Main"

b It will print "Main returns"

c It will not print "Main returns"

d The program will never exit.

e main() method will never return

Q13.

What is guaranteed by the JVM about the method yield()?

Select 1 correct option.

a All lower priority threads will be granted CPU time.

b The current thread will sleep for some time giving a chance to other threads to run.

c Atleast one thread having lower priority will get CPU time.

d The thread will sleep until it is notified.

e Nothing is guaranteed.

Q14.

Which of these statements are true?

Select 2 correct options

a Calling the method sleep( ) does not kill a thread.

b A thread dies when the start( ) method ends.

c A thread dies when the thread's constructor ends.

d A thread dies when the run( ) method ends.

e Calling the method kill( ) stops and kills a thread.

Q15.

Consider the following situation:

Thread T1 holds the lock for an object Obj. Thread T2 is has called obj.wait() and is blocked.

What will allow the thread T2 to become runnable?

Select 1 correct option.

a Thread T1 calls Thread.sleep(100).

b Thread T2's wait() times out.

c Thread T1 is interrupted.

d Thread T1 releases the lock on Obj and calls the notify() method on T2.

e Thread T1 releases the lock on Obj and calls the notify() method on Obj.