PROGRAMMING IN JAVA

Assignment6

TYPE OF QUESTION: MCQ

Number of questions: 15 **Total mark: 15 × 1 = 15**

QUESTION 1:

Which of the following is not suitable in a single CPU environment?

a. Multiprogramming
b. Multithreading
c. Multiprocessing
d. Multitasking

Correct Answer: c

Detailed Solution:
All are concurrent programming in a single CPU environment in the sense that execution of multiple processes are taken place except the multiprocessing where a process is executed with multiple processors.

QUESTION 2:

Which class/ interface should be considered to create a Java program to define multiple threads?

a. Thread
b. Runnable
c. Exception
d. No class/ interface is required to write a thread program.

Correct Answer: a, b

Detailed Solution:
There are two ways that a thread program can be defined in Java: Using the class Thread and interface Runnable. These two things are defined in `java.lang` package.

QUESTION 3:

What is/ are of the following method(s) necessary to be called to start the execution of a thread?

a. init()

b. start()
c. run( )
d. None of the above as automatically a thread program starts it execution.

Correct Answer:b

Detailed Solution:
There are is no init() method in the context of multithreaded programming in Java. The run() method is no need to be defined in order to define a thread.

QUESTION 4:

Which will contain the body of the thread?

a. run();
b. start();
c. stop();
d. main();

Correct Answer: a

Detailed Solution:
The run() method to a thread is like the main() method to an application. Starting the thread causes the object's run method to be called in that separately executing thread**.**

QUESTION 5:

Let’s consider the following program in Java.

```
class Test extends Thread {
publicvoid run() {
       System.out.println("I am from Run…");
}
}

class MyProgram {
public  static void main(String[] args) {
       Test t = new Test();
       t.start();
  }
}
```

If you run this program the how many threads will be executed altogether?

a. One thread only.
b. Two threads only.
c. Three threads only.
d. No thread will run in this case.

Correct Answer: b

Detailed Solution:
Here, two thread objects will be in execution: One is the thread due to the execution of the `main()` method and other is the `run()` of the object t.

QUESTION 6:

A thread is defined using the java.lang.Thread class, which is as follows.

```
public class MyThreadextendaThread{

    public void run()     {
        // Some code here ….
    }
}
```

Which of the following statement will require to create a thread and then start running it?

```
a. MyThread t = new MyThread();
   t.start();
b. MyThread t = new MyThread();
   t.run();
c. MyThread t;
   t.start();
d. MyThread t;
   t.run();
```

Correct Answer: a

Detailed Solution:
The standard procedure of running a thread using `class Thread`is to define a thread class (e.g., here, the `MyThread`as a sub class of the `class Thread`) then in a `main()`, create a thread object of class `MyThread`and then call the `start()` method of the thread object.

QUESTION 7:

A thread is defined using the java.lang.Runnable interface, which is as follows.

```
public class MyThreadimplementsRunnable{

    public void run()     {
        // Some code here ….
    }
}
```

Which of the following statement will require to create a thread and then start running it?

```
a. MyThread t = new MyThread();
   t.start();
b. MyThread t = new MyThread();
   t.run();
c. Thread t = new Thread(MyThread);
   t.start();
d. Thread t = new Thread(MyThread);
   t.run();
```

Correct Answer: c

Detailed Solution:
The standard procedure of running a thread using `Runnable` interface is to create a class implementing the run() (e.g., here, the `MyThreadimplementsRunnable`) then in a `main()`, create a thread object of class `Thread` and then call the `start()` method of the thread object passing the class as an argument and then call the `start()` of the class `Thread`.

QUESTION 8:

What will happen if two threads of the different priority values are called to run simultaneously?

a. The thread for which the `start()` is called first will start its execution first irrespective of their priority.

b. The thread with higher priority value will start its execution first, completes its execution and then the thread with lower priority will start.
c. The thread with lower priority value will start its execution first, completes its execution and then the thread with higher priority will start.
d. The thread with higher priority value will start its execution first, and then the thread with lower priority will start. Subsequently, both the threads will run simultaneously.

Correct Answer:d

Detailed Solution:
When two threads are started from the `main()` method one after another. say, then the thread with higher priority value will start first and then the thread with lower priority and subsequently both the threads share the CPU simultaneously.

QUESTION 9:

How many priory values that a thread can be assigned?

a. 2
b. 3
c. Any number of values within any range.
d. Any number of values within the range of 1 to 10 both inclusive;

Correct Answer: b

Detailed Solution:
There are three priority values that a thread can be assigned: MIN_PRIRITY with the value 1, NORM_PRIRITY with the value 5, and MAX_PRIRITY with the value 10. If no priority value is assigned, then the default priority value is NORM_PRIRITY.

QUESTION 10:

What will happen if two threads of the same priority value are called to run simultaneously?

a. Any one thread can start its execution with 50-50 chance for each.
b. The thread for which the `start()` is called first will start its execution first.
c. The order of execution is decided by the operating system (OS) of the system where the program is running.

d. The thread whose run method is defined first in the program will be executed first.

Correct Answer: a

Detailed Solution:
When two threads with the same priority values are started from the `main()` method,then any one can starts its execution first irrespective of the order their `start()` methods are called or their `run()` are defined.

QUESTION 11:

Consider the execution of the following programs.

```
class Thread1 extends Thread {
    public void run()   {
        for(inti = 0; i< 3; i++)  {
System.out.print("A");
System.out.print("B");
        }
    }
}

class Thread2 extends Thread  {
    public void run()     {
        for(inti = 0; i< 3; i++)          {
System.out.print("C");
System.out.print("D");
        }
    }
}

class MainThread {
       public static void main(String args[]) {
              Thread1 t1 = new Thread1();
              Thread2 t2 = new Thread2();
              t1.start();
              t2.start();
    }
}
```

What will be the output likely to be?

a. Will print in this order ABCDABCD...ABCD
b. Will print in this order ACBDACBD...ACBD
c. Will print CDABCDAB...CDAB
d. Will print in any random order of A, B, C, D which is in fact different in different runs.

Correct Answer: d

Detailed Solution:
The order of output cannot be predicted, it can be in any order and the order is also not same form one run to another.

QUESTION 12:

Which of the following are the valid constructor(s) of `Thread` class?

```
  i.   Thread(Runnable r, String name;)
 ii.   Thread( );
iii.   Thread(int priority);
 iv.   Thread(Runnable r, ThreadGroup g);
  v.   Thread(Runnable r, int priority);
```

a. i and ii
b. ii and iv
c. i and iii
d. ii and v

Correct Answer: a

Detailed Solution:
At the time of creating a thread object, neither its priority value nor its group can be set. All these setting are possible after the creation of thread objects only.

QUESTION 13:

Which of the following methods make thread leave the running state?

```
  i.   yield()
 ii.   wait()
iii.   notify()
 iv.   notifyAll()
  v.   sleep(1000)
 vi.   join()
vii.   suspend()
```

a. iii, iv and vii
b. i, ii and iv
c. i, ii, v, vi, vii
d. i, iv and vii

Correct Answer: c

Detailed Solution:

`notify()` and `notifyALL()` methods changes the state of thread from blocked to runnable states.

QUESTION 14:

Consider the execution of the following programs.

```
class Thread1 extends Thread {
synchronized(this) {
    public void run()   {
        for(inti = 0; i< 3; i++)  {
System.out.print("A");
System.out.print("B");
        }
    }
  }
}

class Thread2 extends Thread  {
synchronized(this) {
    public void run()     {
        for(inti = 0; i< 3; i++)  {
System.out.print("C");
System.out.print("D");
        }
    }
  }
}

class MainThread {
       public static void main(String args[]) {
              Thread1 t1 = new Thread1();
              Thread2 t2 = new Thread2();
              t1.start();
              t2.start();
    }
}
```

What will be the output likely to be when the we synchronized the execution of the threads?

a. Will print in this order ABCDABCD...ABCD
b. Will print in this order ACBDACBD...ACBD
c. Will print CDABCDAB...CDAB
d. Will print in any random sequence of AB, CD.

Correct Answer: d

Detailed Solution:

There will be pairs of AB and CDm and their order of appearance cannot be predicted, it can be in any order and the order is also not same form one run to another.

QUESTION 15:

Let us consider the following program segements.

```
class ThreadDemo implements Runnable {
    String x, y;
    public void run()     {
        for(inti = 0; i< 10; i++)
            synchronized(this)
            {
                x = “Hello”;
                y = “Java”;
System.out.print(x + " " + y + " ");
            }
    }

    public static void main(String args[])     {
ThreadDemo run = new ThreadDemo ();
        Thread obj1 = new Thread(run);
        Thread obj2 = new Thread(run);
        obj1.start();
        obj2.start();
    }
}
```

a. There will be an arbitrary order the two strings Hello and Java 10 times each.
b. It will print Hello Java …. 20 times.
c. It will print Hello 10 times and then Java 10 times.
d. This program will not print anything.

Correct Answer: b

Detailed Solution:
A block with synchronized keyword is atomic, that is, when control enters into the block, it will not allow any other block in other threads to execute. Note that, here two threads namely `obj1` and `obj2` execute two loops (that is, two blocks of codes) simultaneously.

***********END************