Multi-Threading

1. From the below code how many threads will be created?

```
class Test extends Thread {
public void run()
   {
      System.out.println("Run");
   }
}
class Myclass {
public static void main(String[] args)
   {
      Test t = new Test();
      t.start();
   }
}
```

Output

1. One thread created
2. Two thread created
3. Depend upon system
4. No thread created

Ans: The answer is option (2)

Explanation : In the above program, one thread will be created i.e. the main thread which is responsible to execute the main() method and the child thread will be created after the execution of t.start() which is responsible to execute run() method.

2. From the below code how many threads will be created?

```
class Test extends Thread {
public void run()
   {
      System.out.println("Run");
   }
}
class Myclass {
public static void main(String[] args)
   {
      Test t = new Test();
      t.run();
   }
}
```

Output

1. One thread created
2. Two thread created
3. Depend upon system
4. No thread created

- The answer is option (1)
- **Explanation :** In the above program only one thread will be created i.e. the main thread which is responsible to execute the main() method only. The run() method is called by the object t like a normal method.

3. Find Output

```
class Test extends Thread {
public void run()
   {
      System.out.print("Run ");
   }
}
class Myclass {
public static void main(String[] args)
   {
      Test t = new Test();
      t.start();
      System.out.print("Main ");
   }
}
```

Output

1. Main Run
2. Run Main
3. Depend upon Program
4. Depend upon JVM

- The answer is option (4)
- **Explanation :** In the above program, we cant predict the exact order of the output as it is decided by the Thread scheduler which is the part of JVM.

4. Find Output

```
class Test implements Runnable {
public void run()
   {
      System.out.print("Run ");
   }
}
class Myclass {
public static void main(String[] args)
   {
      Test t = new Test();
      t.start();
      System.out.print("Main ");
   }
}
```

Output

1. Main Run
2. Run Main
3. Compile time error
4. Depend upon JVM

- The answer is option (3)
- **Explanation :** In the above program, we will get compile time error because start() method is present in the Thread class only and we are implementing Runnable interface.

5. Find Output

```
class Test implements Runnable {
public void run()
   {
      System.out.print("Run ");
   }
}
class Myclass {
public static void main(String[] args)
   {
      Thread t1 = new Thread();
      t1.start();
      System.out.print("Main ");
   }
}
```

Output

1. Run
2. Main
3. Compile time error
4. Run Main

- The answer is option (2)
- **Explanation :** In the above program, we are calling start() method of Thread class which is responsible to execute run() method of Thread class and Thread class run() method has empty implementation. That’s why one child thread will be created but it will not execute Test class run() method.

6. Find Output

```
public class Test implements Runnable
{
   public void run()
   {
      System.out.printf("Java ");
   }
   public static void main(String[] args) throws InterruptedException
   {
      Thread thread1 = new Thread(new Test());
      thread1.start();
      thread1.start();
      System.out.println(thread1.getState());
   }
}
```

Output

- a) Java Java TERMINATED
  b) Java TERMINATED
  c) Compilation Error
  d) Runtime Error
- **Ans.** (d)
  Explanation: Invoking start() method on a thread moves the thread to a RUNNABLE state. But invoking start() method on a thread that has already started throws a IllegalThreadStateException because the thread is already in RUNNABLE state.

7. Find Output

```
public class Test extends Thread implements Runnable
{
    public void run()
    {
        System.out.printf("Java ");
    }
    public static void main(String[] args) throws InterruptedException
    {
        Test obj = new Test();
        obj.run();
        obj.start();
    }
}
```

Output

a) Runtime error
b) Compilation error
c) Java Java
d) Java

Ans. (c)
Explanation: Test class extends Thread class that has start() method implemented. So invoking start() on an object that extends Thread class invokes run() method defined in the program.

8. Find Output

Which of these method of Thread class is used to find out the priority given to a thread?
a) get()
b) ThreadPriority()
c) getPriority()
d) getThreadPriority()

Output

Answer: c

9 .Find Output

```
public class Test
{
   public static void main(String[] args) throws
InterruptedException
   {
   Thread t = Thread.currentThread();
      t.setName("New Thread");
      System.out.println(t);
   }
}
```

Output

a) Thread[5,main]
b) Thread[New Thread,5]
c) Thread[main,5,main]
d) Thread[New Thread,5,main]

Answer: d

10. Find Output

```
public class Test
{
   public static void main(String[] args) throws
InterruptedException
   {
   Thread t = Thread.currentThread();
      t.setName("New Thread");
      System.out.println(t.getName());
   }
}
```

Output

a) main
b) Thread
c) New Thread
d) Thread[New Thread,5,main]

- Answer: c
  Explanation: The getName() function is used to obtain the name of the thread, in this code the name given to thread is ‘New Thread’.

11. Find Output

```
public class Test
{
   public static void main(String[] args) throws
InterruptedException
   {
   Thread t = Thread.currentThread();
      System.out.println(t.getPriority());
   }
}
```

Output

a) 0
b) 1
c) 4
d) 5

Answer: d

12. Find Output

```
public class Test
{
   public static void main(String[] args) throws
InterruptedException
   {
   Thread t = Thread.currentThread();
   System.out.println(t.isAlive());
   }
}
```

Output

a) 0
b) 1
c) true
d) false

Answer: c
Explanation: Thread t is seeded to currently program, hence when you run the program the thread becomes active & code ‘t.isAlive’ returns true.

13. Find Output

What is maximum thread priority in Java

a) 10

b) 12

c) 5

d) 8

Output

Answer: A

14. Find Output

```
class UserThread extends Thread {
public UserThread() {
super("my thread");
start();
}

public void run() {
System.out.print("hello ");
}
}

public class Test {
public static void main(String[] args) {
UserThread u = new UserThread();
UserThread u1 = new UserThread();
}
}
```

Output

a) hello hello
b) Hello
c) Compilation error
d) Exception

Ans: a