Tech Mahindra

Java – Learn From Home

Quiz – Chapter 4

1. What will be the output of the program?

```
public class Foo
{
   public static void main(String[] args)
   {
      try
      {
         return;
      }
      finally
      {
         System.out.println( "Finally" );
      }
   }
}
```

A. Finally

B. Compilation fails.

C. The code runs with no output.

D. An exception is thrown at runtime.

Answer: Option A

Explanation:
If you put a finally block after a try and its associated catch blocks, then once execution enters the try block, the code in that finally block will definitely be executed except in the following circumstances:
An exception arising in the finally block itself.
The death of the thread.

The use of System.exit()
Turning off the power to the CPU.
It suppose the last three could be classified as VM shutdown.

2. What will be the output of the program?

```
try
{
   int x = 0;
   int y = 5 / x;
}
catch (Exception e)
{
   System.out.println("Exception");
}
catch (ArithmeticException ae)
{
   System.out.println(" Arithmetic Exception");
}
System.out.println("finished");
```

A. finished

B. Exception

C. Compilation fails.

D. Arithmetic Exception

Answer: Option C

Explanation:
Compilation fails because ArithmeticException has already been caught. ArithmeticException is a subclass of java.lang.Exception, by time the ArithmeticException has been specified it has already been caught by the Exception class.
If ArithmeticException appears before Exception, then the file will compile. When catching exceptions the more specific exceptions must be listed before the more general (the subclasses must be caught before the super classes)

3. What will be the output of the program?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod();
         System.out.print("A");
      }
      catch (Exception ex)
      {
         System.out.print("B");
      }
      finally
      {
         System.out.print("C");
      }
      System.out.print("D");
   }
   public static void badMethod()
   {
      throw new Error(); /* Line 22 */
   }
}
```

A. ABCD

B. Compilation fails.

C. C is printed before exiting with an error message.

D. BC is printed before exiting with an error message.

Answer: Option C

Explanation:
Error is thrown but not recognised line (22) because the only catch attempts to catch an Exception and Exception is not a superclass of Error. Therefore only the code in the finally statement can be run before exiting with a runtime error (Exception in thread "main" java.lang.Error).

4. What will be the output of the program?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod();
         System.out.print("A");
      }
      catch (RuntimeException ex) /* Line 10 */
      {
         System.out.print("B");
      }
      catch (Exception ex1)
      {
         System.out.print("C");
      }
      finally
      {
         System.out.print("D");
      }
      System.out.print("E");
   }
   public static void badMethod()
   {
      throw new RuntimeException();
   }
}
```

A. BD

B. BCD

C. BDE

D. BCDE

Answer: Option C

Explanation:
A Run time exception is thrown and caught in the catch statement on line 10. All the code after the finally statement is run because the exception has been caught

5. What will be the output of the program?

```
public class RTExcept
{
   public static void throwit ()
   {
      System.out.print("throwit ");
      throw new RuntimeException();
   }
   public static void main(String [] args)
   {
      try
      {
         System.out.print("hello ");
         throwit();
      }
      catch (Exception re )
      {
         System.out.print("caught ");
      }
      finally
      {
         System.out.print("finally ");
      }
```

```
        System.out.println("after ");
    }
}
```

A. hello throwit caught

B. Compilation fails

C. hello throwit RuntimeException caught after

D. hello throwit caught finally after

Answer: Option D

Explanation:
The main() method properly catches and handles the RuntimeException in the catch block, finally runs (as it always does), and then the code returns to normal. A, B and C are incorrect based on the program logic described above. Remember that properly handled exceptions do not cause the program to stop executing

6. What will be the output of the program?

```
public class Test
{
   public static void aMethod() throws Exception
   {
      try /* Line 5 */
      {
         throw new Exception(); /* Line 7 */
      }
      finally /* Line 9 */
      {
         System.out.print("finally "); /* Line 11 */
      }
   }
   public static void main(String args[])
   {
      try
      {
```

```
        aMethod();
      }
      catch (Exception e) /* Line 20 */
      {
        System.out.print("exception ");
      }
      System.out.print("finished"); /* Line 24 */
    }
}
```

A. finally

B. exception finished

C. finally exception finished

D. Compilation fails

Answer: Option C

Explanation:
This is what happens:
(1) The execution of the try block (line 5) completes abruptly because of the throw statement (line 7).
(2) The exception cannot be assigned to the parameter of any catch clause of the try statement therefore thefinally block is executed (line 9) and "finally" is output (line 11).
(3) The finally block completes normally, and then the try statement completes abruptly because of the throwstatement (line 7).
(4) The exception is propagated up the call stack and is caught by the catch in the main method (line 20). This prints "exception".
(5) Lastly program execution continues, because the exception has been caught, and "finished" is output (line 24)

7. What will be the output of the program?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod();
         System.out.print("A");
      }
      catch (Exception ex)
      {
         System.out.print("B");
      }
      finally
      {
         System.out.print("C");
      }
      System.out.print("D");
   }
   public static void badMethod() {}
}
```

A. AC

B. BC

C. ACD

D. ABCD

Answer: Option C

Explanation:
There is no exception thrown, so all the code with the exception of the catch statement block is run

8. What will be the output of the program?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod(); /* Line 7 */
         System.out.print("A");
      }
      catch (Exception ex) /* Line 10 */
      {
         System.out.print("B"); /* Line 12 */
      }
      finally /* Line 14 */
      {
         System.out.print("C"); /* Line 16 */
      }
      System.out.print("D"); /* Line 18 */
   }
   public static void badMethod()
   {
      throw new RuntimeException();
   }
}
```

A. AB

B. BC

C. ABC

D. BCD

Answer: Option D

Explanation:
(1) A RuntimeException is thrown, this is a subclass of exception.
(2) The exception causes the try to complete abruptly (line 7) therefore line 8 is never executed.
(3) The exception is caught (line 10) and "B" is output (line 12)
(4) The finally block (line 14) is always executed and "C" is output (line 16).
(5) The exception was caught, so the program continues with line 18 and outputs "D

9. What will be the output of the program?

```
public class MyProgram
{
   public static void main(String args[])
   {
      try
      {
         System.out.print("Hello world ");
      }
      finally
      {
         System.out.println("Finally executing ");
      }
   }
}
```

A. Nothing. The program will not compile because no exceptions are specified.

B. Nothing. The program will not compile because no catch clauses are specified.

C. Hello world.

D. Hello world Finally executing

Answer: Option D

Explanation:
Finally clauses are always executed. The program will first execute the try block, printing Hello world, and will then execute the finally block, printing Finally executing.
Option A, B, and C are incorrect based on the program logic described above. Remember that either a catch or a finally statement must follow a try. Since the finally is present, the catch is not required

10. What will be the output of the program?

```
class Exc0 extends Exception { }
class Exc1 extends Exc0 { } /* Line 2 */
public class Test
{
   public static void main(String args[])
   {
      try
      {
         throw new Exc1(); /* Line 9 */
      }
      catch (Exc0 e0) /* Line 11 */
      {
         System.out.println("Ex0 caught");
      }
      catch (Exception e)
      {
         System.out.println("exception caught");
      }
   }
}
```

A. Ex0 caught

B. exception caught

C. Compilation fails because of an error at line 2.

D. Compilation fails because of an error at line 9.

Answer: Option A

Explanation:
An exception Exc1 is thrown and is caught by the catch statement on line 11. The code is executed in this block. There is no finally block of code to execute