DAY00

What is the advantage of using exception handling?

Exception enables a method to throw an exception to its called. The caller can handle this exception. (Without this capability, the called method itself must handle the exception or terminate the program.)

Which of the following statements will throw an exception?

System.out.println(1 / 0);

System.out.println(1.0 / 0);

The statement
System.out.println(1 / 0); will throw a divide by zero exception

Point out the problem in the following code. Does the code throw any exceptions?

long value = Long.MAX_VALUE + 1;

System.out.println(value);

The problem will be in the override and the code will not throw an exception.

What does the JVM do when an exception occurs? How do you catch an exception?

When an exception occurs, the JVM searches for catch clause related to that exception.
We can use the try-catch statement to catch an exception as the following syntax:

```
try
{
// throws the exception if occurs
}
catch (Exception e)
{
```

// catch and process the exception
}

What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    try {
      int value = 30;
      if (value < 40)
        throw new Exception("value is too small");
    }
    catch (Exception ex) {
      System.out.println(ex.getMessage());
    }
    System.out.println("Continue after the catch block");
  }
}
```

Output 30:

value is too small
Continue after the catch block

Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < 2; i++) {
      System.out.print(i + " ");
      try {
        System.out.println(1 / 0);
      }
      catch (Exception ex) {
      }
    }
  }
}
```

(a)

```
public class Test {
  public static void main(String[] args) {
    try {
      for (int i = 0; i < 2; i++) {
        System.out.print(i + " ");
        System.out.println(1 / 0);
      }
    }
    catch (Exception ex) {
    }
  }
}
```

(b)

a. Output: 0 1
b. Output: 0

What is a checked exception, and what is an unchecked exception?

Checked exception: Checked exception is called as compile time exception. It must be declared explicitly in the method header and must be generated in the method.

Unchecked exception: Unchecked exception is also called as run time exception. It needs not to be declared explicitly in the method header.

Suppose that *statement2* causes an exception in the following try-catch block: Answer the following questions:

Will *statement3* be executed?

If the exception is not caught, will *statement4* be executed?

If the exception is caught in the catch block, will *statement4* be executed?

```
try {
        statement1;
        statement2;
        statement3;
}
catch (Exception1 ex1) {
}
catch (Exception2 ex2) {
}
statement4;
```

The statement3 is not executed.
Since statement2 causes an exception which transfers the control to catch block.

The statement4 is not executed if the exception is not caught.

Only the statements in the try block will be executed if no statement int he try block causes an exception.

The statement4 is executed if the exception is caught in the catch block.

Correct a compile error in the following code:

```
public void m(int value) {
        if (value < 40)
                throw new Exception("value is too small");
```

In the method header the declaration of Exception is not specified. The Exception in a method must be declared in its header using the key word throws before using throw in its body.

Modified Code:

```
public void m(int value) throws Exception
{
if (value < 40)
throw new Exception("value is too small");
}
```