www.Javagenious.com

Free Downloads From www.Javagenious.com

10/10/2011

info@javagenious.com

Q:Does Java provide any construct to find out the size of an object?

A: No there is not sizeof operator in Java. So there is not direct way to determine the size of an object directly in Java.

Q:Give a simplest way to find out the time a method takes for execution without using any profiling tool?

A: Read the system time just before the method is invoked and immediately after method returns. Take the time difference, which will give you the time taken by a method for execution.

To put it in code...

```
long start = System.currentTimeMillis ();
method ();
long end = System.currentTimeMillis ();
System.out.println ("Time taken for execution is " +
(end - start));
```

Remember that if the time taken for execution is too small, it might show that it is taking zero milliseconds for execution. Try it on a method which is big enough, in the sense the one which is doing considerable amout of processing.

Q:What are wrapper classes?

A: Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes. They are e.g. Integer, Character, Double etc.

Q:Why do we need wrapper classes?

A: It is sometimes easier to deal with primitives as objects. Moreover most of the collection classes store objects and not primitive data types. And also the wrapper classes provide many utility methods also. Because of these resons we need wrapper classes. And since we create instances of these classes we can store them in any of the collection classes and pass them around as a collection. Also we can pass them around as method parameters where a method expects an object.

Q:What are checked exceptions?

A: Checked exception are those which the Java compiler forces you to catch. e.g. IOException are checked Exceptions.

Q:What are runtime exceptions?

A: Runtime exceptions are those exceptions that are thrown at runtime because of either wrong input data or because of wrong business logic etc. These are not checked by the compiler at compile time.

Q:What is the difference between error and an

exception?

A: An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors and you can not repair them at runtime. While exceptions are conditions that occur because of bad input etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving user a feedback for entering proper values etc.).

Q:How to create custom exceptions?

A: Your class should extend class Exception, or some more specific type thereof.

Q:If I want an object of my class to be thrown as an exception object, what should I do?

A: The class should extend from Exception class. Or you can extend your class from some more precise exception type also.

Q:If my class already extends from some other class what should I do if I want an instance of my class to be thrown as an exception object?

A: One can not do anytihng in this scenarion. Because Java does not allow multiple inheritance and does not provide any exception interface as well.

Q:How does an exception permeate through the code?

A: An unhandled exception moves up the method stack in search of a matching When an exception is thrown from a code which is wrapped in a try block followed by one or more catch blocks, a search is made for matching catch block. If a matching type is found then that block will be invoked. If a matching type is not found then the exception moves up the method stack and reaches the caller method. Same procedure is repeated if the caller method is included in a try catch block. This process continues until a catch block handling the appropriate type of exception is found. If it does not find such a block then finally the program terminates.

Q:What are the different ways to handle exceptions?

A: There are two ways to handle exceptions,

1. By wrapping the desired code in a try block followed by a catch block to catch the exceptions.
and
2. List the desired exceptions in the throws clause of the method and let the caller of the method hadle those exceptions.

Q:What is the basic difference between the 2 approaches to exception handling.
1> try catch block and
2> specifying the candidate exceptions in the throws clause?
When should you use which approach?

A: In the first approach as a programmer of the method, you urself are dealing with the exception. This is fine if you are in a best position to decide should be done in case of an exception. Whereas if it is not the responsibility of the method to deal with it's own exceptions, then do not use this approach. In this case use the second approach. In the second approach we are forcing the caller of the method to catch the exceptions, that the method is likely to throw. This is often the approach library creators use. They list the exception in the throws clause and we must catch them. You will find the same approach throughout the java libraries we use.

Q:Is it necessary that each try block must be followed by a catch block?

A: It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

Q:If I write return at the end of the try block, will the finally block still execute?

A: Yes even if you write return as the last statement in the try block and no exception occurs, the finally block will execute. The finally block will execute and then the control return.

Q:If I write System.exit (0); at the end of the try block, will the finally block still execute?

A: No in this case the finally block will not execute because when you say System.exit (0); the control immediately goes out of the program, and thus finally never executes.