**Question 1**

What is the Output for the following code snippet?

**public** **class** test {

**public** **static** **void** main(String[] args) {

**int** i = 42;

String s = (i<40)?"life":(i>50)?"universe":"everything";

System.*out*.println(s);

}

}

Solution: everything

**Question 2**

What are the two types of exceptions? Explain why we need them?

Solution: Checked and unchecked exceptions.

Unchecked Exception: These are runtime exceptions which represent conditions that, generally speaking reflect errors in the programs logic. They are subclasses of RuntimeException and are usually implemented using [IllegalArgumentException](http://java.sun.com/javase/6/docs/api/java/lang/IllegalArgumentException.html), [NullPointerException](http://java.sun.com/javase/6/docs/api/java/lang/NullPointerException.html), or [IllegalStateException](http://java.sun.com/javase/6/docs/api/java/lang/IllegalStateException.html).

Checked Exception: Exceptions that can be handled and thrown or caught are checked exception.

**Question 3**

What is the difference between Arrays.sort() vs Collections.sort()? Which one is faster?

**Question 4**

Explain reflection with an example?

Solution: Reflection is a feature which allows a java program to examine or introspect upon itself and manipulate internal properties of the program.

Example: Program to obtain the names of all its members and display them

import java.lang.reflect.\*;

public class DumpMethods {

public static void main(String args[])

{

try {

Class c = Class.forName(args[0]);

Method m[] = c.getDeclaredMethods();

for (int i = 0; i < m.length; i++)

System.out.println(m[i].toString());

}

catch (Throwable e) {

System.err.println(e);

}

}

}

**Question 5**

What is the difference between >> and >>> operators?

**Question 6**

What is the purpose of finalize() method?

Solution: Every class inherits the finalize method from object. The method is called by the garbage collector when it determines no more references to the object exists. The finalize method performs no actions but it may be overridden by any class. Example usage would be to clean up non-java resources ie closing a file.

**Question 7**

Explain Serialization and deserialization?

Solution: Serialization is the process of converting a set of object instances that contain references to each other into a linear stream of bytes, which can then be sent through a socket or stored in a file.

The serialized stream of data can be programmatically manipulated, and a deep copy of the objects can be made by reversing the process. This reversal process is called deserialization.

**Question 8**

What is the difference between sleep and wait?

Solution: Thread.sleep sends the current thread into the “Not Runnable” state for some amount of time. The thread keeps the monitors it has acquired. If the thread is currently in a synchronized block or method no other thread can enter this block or method.

**Question 9**

What is the difference between static and non-static variables?

Solution: A static variable is associated with the class as a whole rather than with specific instancesof a class. Non-static variables take on unique values with each object instance.

**Question 10**

Provide the output for the program

**public** **class** Animal {

**public** **static** **void** main(String[] args) {

String pig = "length: 10";

String dog = "length: " + pig.length();

System.*out*.println("Animals are equal: " + pig == dog);

}

}

Output: false

**Question 11**

What is the difference between GET and POST method?

Solution: The fundamental difference between *METHOD="GET"* and *METHOD="POST"* is that they correspond to **different HTTP requests**, as defined in the [HTTP](http://www.w3.org/Protocols/) [specifications](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9). The submission process for both methods begins in the same way - a [form data set](http://www.w3.org/TR/REC-html40/interact/forms.html#form-data-set) is constructed by the browser and then encoded in a manner specified by the *enctype* attribute. For *METHOD="POST* the *enctype* attribute can be *multipart/form-data* or *application/x-www-form-urlencoded*, whereas for *METHOD="GET"*, only *application/x-www-form-urlencoded* is allowed. This form data set is then transmitted to the [server](http://www.diffen.com/difference/Get_vs_Post" \t "undefined)

[http://konac.kontera.com/javascript/lib/imgs/grey_loader.gif](http://www.diffen.com/difference/Get_vs_Post" \t "undefined)

For form submission with METHOD="GET", the browser constructs a URL by taking the value of the *action* attribute, appending a *?* to it, then appending the form data set (encoded using the application/x-www-form-urlencoded content type). The browser then processes this URL as if following a link (or as if the user had typed the URL directly). The browser divides the URL into parts and recognizes a host, then sends to that host a GET request with the rest of the URL as argument. The server takes it from there. Note that this process means that the form data are restricted to ASCII codes. Special care should be taken to encode and decode other types of characters when passing them through the URL in ASCII format.

Submission of a form with METHOD="POST" causes a POST request to be sent, using the value of the *action* attribute and a message created according to the content type specified by the *enctype* attribute.

**Pros and Cons**

Since form data is sent as part of the URL when *GET* is used --

* Form data are restricted to ASCII codes. Special care should be taken to encode and decode other types of characters when passing them through the URL in ASCII format. On the other hand, binary data, images and other files can all be submitted through *METHOD="POST"*
* All form data filled in is visible in the URL. Moreover, it is also stored in the user's web browsing history/logs for the browser. These issues make *GET* less secure.
* However, one advantage of form data being sent as part of the URL is that one can bookmark the URLs and directly use them and completely bypass the form-filling process.
* There is a limitation on how much form data can be sent because URL lengths are limited.

**Question 12**

What is the difference between static variables and instance variables in a servlet?

Solution: According to the Java Language definition, a static variable is shared among all instances of a class, where a non-static variable -- also called an instance variable -- is specific to a single instance of that class.

According to the Servlet specification, a servlet that does not declare SingleThreadModel usually has one and only one instance, shared among all concurrent requests hitting that servlet.

That means that, in servlets (and other multithreaded applications), an instance variable behaves very much like a static variable, since it is shared among all threads. You have to be very careful about synchronizing access to shared data.

**Question 13**

What is the Difference between client side vs server side validation? Explain.

Solution: Client side validation can be done using javascript, server side validation can be done in actionforms or servlets.

**Question 14**

Write a program that provides the nth number in the following series and also the function should return he nth number in the series

F(n) = F(n-1) + F(n-2)

Solution:

public static int fib(int n) { if (n <= 2)

return 1;

else

return fib(n - 1) + fib(n - 2);

}

**Question 15**

Write program/pseudocode to wrap long input lines into two or more shorter lines after the last non-blank character that occurs before the n-th column of the input. Make sure your program does something intelligent with very long lines and if there are no blanks or tabs before the specified column.

Solution:

**#include <stdio.h>**

**#define MAXLINE 1000 /\* max input line size \*/**

**char** line[MAXLINE]; /\*current input line\*/

**int** getline(**void**); /\* taken from the KnR book. \*/

**int**

main()

{

**int** t,len;

**int** location,spaceholder;

**const** **int** FOLDLENGTH=**70**; /\* The max length of a line \*/

**while** (( len = getline()) > **0** )

{

**if**( len < FOLDLENGTH )

{

}

**else**

{

/\* if this is an extra long line then we

\*\* loop through it replacing a space nearest

\*\* to the foldarea with a newline.

\*/

t = **0**;

location = **0**;

**while**(t<len)

{

**if**(line[t] == **' '**)

spaceholder = t;

**if**(location==FOLDLENGTH)

{

line[spaceholder] = **'\n'**;

location = **0**;

}

location++;

t++;

}

}

printf ( "%s", line);

}

**return** **0**;

}

/\* getline: specialized version \*/

**int** getline(**void**)

{

**int** c, i;

**extern** **char** line[];

**for** ( i=**0**;i<MAXLINE-**1** && ( c=getchar()) != EOF && c != **'\n'**; ++i)

line[i] = c;

**if**(c == **'\n'**)

{

line[i] = c;

++i;

}

line[i] = **'\0'**;

**return** i;

}

**Question 16**

Which is faster: finding an item in a hash table or in a sorted list?

Solution: Item retrieval is basically O(1) in a hash table, while O(log n) in a sorted list, so the hash table is faster on average.

**Question 17**

Given the following Employee table

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data type** | **Nullable** |
| EMP\_ID | NUMBER(4) | NOT NULL |
| LAST\_NAME | VARCHAR(30) | NOT NULL |
| FIRST\_NAME | VARCHAR(30) |  |
| DEPT\_ID | NUMBER(2) |  |
| JOB\_CAT | VARCHARD2(30) |  |
| SALARY | NUMBER(8,2) |  |

Write the sql query to show the maximum salary paid in each job category of each department?

Solution: Shows the department ID, minimum salary, and maximum salary paid in the department, only is the minimum salary is less than 5000 and the maximum salary is more than 15000

**Question 18**

Why do we need Ajax? Describe the role of XMLHttpRequest/Response object?

Solution: To avoid refresh of the web page and improvise the usability of a system ajax can be used.

*XMLHttpRequest (XHR) is an* [*API*](http://en.wikipedia.org/wiki/Application_programming_interface) *available in* [*web browser*](http://en.wikipedia.org/wiki/Web_browser)[*scripting languages*](http://en.wikipedia.org/wiki/Scripting_language) *such as* [*JavaScript*](http://en.wikipedia.org/wiki/JavaScript)*. It is used to send* [*HTTP*](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) *or* [*HTTPS*](http://en.wikipedia.org/wiki/HTTP_Secure) *requests directly to a* [*web server*](http://en.wikipedia.org/wiki/Web_server) *and load the* [*server response*](http://en.wikipedia.org/wiki/Response) *data directly back into the script.*[*[1]*](http://en.wikipedia.org/wiki/XMLHttpRequest#cite_note-0) *The data might be received from the server as* [*XML*](http://en.wikipedia.org/wiki/XML) *text[[2]](http://en.wikipedia.org/wiki/XMLHttpRequest" \l "cite_note-1) or as* [*plain text*](http://en.wikipedia.org/wiki/Plain_text)*.*[*[3]*](http://en.wikipedia.org/wiki/XMLHttpRequest#cite_note-2) *Data from the response can be used directly to alter the* [*DOM*](http://en.wikipedia.org/wiki/Document_Object_Model) *of the currently active document in the browser window without loading a new* [*web page*](http://en.wikipedia.org/wiki/Web_page) *document.*

**Question 19**

How is a session maintained in a web application?

Solution: Applications with EJB use stateful session beans, using httpSession, hidden form fields, url rewriting, cookies

**Question 20**

What is the difference between Hashtable and Hashmap? Can/cannot a HashMap be synchronized? Explain?

Solution: Both provide key-value access to data. The Hashtable is one of the original collection classes in Java. HashMap is part of the new Collections Framework, added with Java 2, v1.2.

The key difference between the two is that access to the Hashtable is synchronized on the table while access to the HashMap isn't. You can add it, but it isn't there by default.

Another difference is that iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn't. If you change the map while iterating, you'll know.

And, a third difference is that HashMap permits null values in it, while Hashtable doesn't.