**Question 1**

Provide the Output for the program

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

**private** test(Object o) {

System.*out*.println("Object");

}

**private** test(Integer dArray) {

System.*out*.println("double array");

}

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

**new** test(**null**);

}

}

Solution: double array

**Question 2**

What is the difference between java.util.Iterator and java.util.ListIterator?

Iterator: An iterator over a collection. Iterator takes the place of Enumeration in the Java collections framework

ListIterator: An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. A ListIterator has no current element; its *cursor position* always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next()

**Question 3**

Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?

Solution: ArrayList and Vector both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The LinkedList is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion. Therefore, the LinkedList allows for fast insertions and deletions.

**Question 4**

Difference between synchronized block vs synchronized method?

Solution: First of all to achieve Multithreading mechanism in java we should go forsynchronization. And this can be done in two ways depending on the requirement.   
  
1. Synchronized block and   
2. Synchronized method.   
  
if you go for synchronized block it will lock a specific object.   
  
if you go for synchronized method it will lock all the objects.   
  
in other way Both the synchronized method and block are used to acquires the lock for an object.

**Question 5**

Explain life cycle of a thread briefly?

1. Solution: **New state –** After the creations of Thread instance the thread is in this state but before the start() method invocation. At this point, the thread is considered not alive.
2. **Runnable (Ready-to-run) state –** A thread start its life from Runnable state. A thread first enters runnable state after the invoking of start() method but a thread can return to this state after either running, waiting, sleeping or coming back from blocked state also. On this state a thread is waiting for a turn on the processor.
3. **Running state –** A thread is in running state that means the thread is currently executing. There are several ways to enter in Runnable state but there is only one way to enter in Running state: the scheduler select a thread from runnable pool.
4. **Dead state –** A thread can be considered dead when its run() method completes. If any thread comes on this state that means it cannot ever run again.
5. **Blocked -** A thread can enter in this state because of waiting the resources that are hold by another thread.

**Question 6**

What is the difference between equals() method and == ?

Solution: Well, first off, == is a fundamental operator in the language. The result type of the expression is a boolean. For comparing boolean types, it compares the operands for the same truth value. For comparing reference types, it compares the operands for the same reference value (i.e., refer to the same object or are both null). For numeric types, it compares the operands for the same integer value or equivalent floating point values. See the Java Language Specification, section 15.20 for more information.

In contrast, equals() is an instance method which is fundamentally defined by the java.lang.Object class. This method, by convention, indicates whether the receiver object is "equal to" the passed in object. The base implementation of this method in the Object class checks for reference equality. Other classes, including those you write, may override this method to perform more specialized equivalence testing. See the Java Language Specification, section 20.1.3 for more information.

The typical "gotcha" for most people is in using == to compare two strings when they really should be using the String class's equals() method. From above, you know that the operator will only return "true" when both of the references refer to the same actual object. But, with strings, most uses want to know whether or not the **value** of the two strings are the same -- since two different String objects may both have the same (or different) values.

**Question 7**

What is the difference between abstract class and an interface and why should we use them?

Solution:

|  |  |
| --- | --- |
| Abstract classes | Interfaces |
| Abstract classes are used only when there is a “is-a” type of relationship between the classes. | Interfaces can be implemented by classes that are not related to one another. |
| You cannot extend more than one abstract class. | You can implement more than one interface |
| Abstract class can implemented some methods also. | Interfaces cannot implement methods. |
| With abstract classes, you are grabbing away each class’s individuality. | With Interfaces, you are merely extending each class’s functionality. |

**Question 8**

Does the order of the catch Statements in a program matter? Explain.

Solution: Yes. IOException is the super class of FileNotFoundException. So, if you put the catch statement for IOException above that for FileNotFoundException, then the code for second catch will become unreachable and the compiler will throw an error for that. Reason is simple: every object of a sub class can be easily accepted by a super class reference.

**Question 9**

What us the difference between prefix and postfix forms of the ++ operator?

Solution: The prefix form performs the increment operation and returns the value of the incrementoperation. The postfix form returns the current value all of the expression and thenperforms the increment operation on that value.

**Question 10**

Provide the result of compiling this program and provide the reason

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

**public** **static** **void** go (Integer n) {System.*out*.println("Integer"); }

**public** **static** **void** go (Integer n) {System.*out*.println("integer"); }

**public** **static** **void** go (Long n) {System.*out*.println("Long"); }

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

Integer y = **new** Integer(6);

Integer z = **new** Integer(7);

*go*(y);

*go*(z);

}

}

Solution: Will not compile. Duplicate method go.

**Question 11**

What is the difference between forward() and sendRedirect()?

**Question 12**

What is a scriplet?

Solution: Scriptlet is code embedded in Jsp page. Default language for scriptlets is Java but it is possible to define other languages as well.

**Question 13**

Difference between static and dynamic include in a jsp?

Solution:

**This is Static include.**

<%@ include file="header.jsp" %>

1.the file includes static text. i.e if we use the static

include, it will merge the contents of included jsp

(header.jsp) with main jsp at compile time. and the size of

the main jsp will increase. Some times it may have problem,

because the maximum size of the jsp is 64kb.

2.if the file is changed,the JSP engine may not recompile

the JSP

**This is Dynamic include.**

<jsp:include page="header.jsp">

1.The file includes Dynamically. i.e at run time.

2.If the included file (header.jsp) is changed,the JSP

engine recompile the included JSP, because it will include

at run time.By using this the jsp size also not increased.

For Example

Static include (<%@ include file="header.jsp" %>) like our

#include (C++)

Dynamic include (<jsp:include page="header.jsp"> ) like our

import (JAVA)

**Question 14**

Write a pseudocode to identify if the expression has extra bracket? (((A+B))\*C))?

Solution:

int fib(int n)

{

int f[n+1];

f[1] = f[2] = 1;

for (int i = 3; i <= n; i++)

f[i] = f[i-1] + f[i-2];

return f[n];

}

**Question 15**

Write a pseudocode for quick sort

Solution:

**procedure Quicksort(left, right:integer);**

index: integer;

if (right - left <= 1) then return;

if (right > left) then

index := Partition(left, right);

Quicksort(left, index-1);

Quicksort(index+1, right);

end Quicksort;

**function Partition(left, right);**

p : integer;

p := A[right];

scan from left until A[i] > p is found;

scan from right until A[j] < p is found;

exchange A[i] and A[j];

continue scanning and exchanging until pointers i, j, cross;

exchange p with A[i];

return (p);

end Partition;

**Question 16**

How would you find a duplicate element in an array of integers between 1 and 1000000. There exists one element which has occurred more than once.

Solution:

Solution1: Pigeon hole technique can be used. Take bool array of size 1,000,000. Elements will be set to false by default. Traverse your array and for every integer in that array keep the bool array element true. Before this check whether that "hole" is already true. If so, it is repeated..

**Question 17**

Examine the description of the 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) |  |

SELECT dept\_id, MIN(salary), MAX(salary)

FROM employees

GROUP BY dept\_id

HAVING MIN(salary) < 5000 AND MAX(salary)

Describe in words what the above sql query solves?

Solution: 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**

Can we use javascript to submit a form?

Yes.

**Question 19**

Evaluate the following query and provide the ouput for the following query

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMPLOYEE\_ID | EMP\_NAME | DEPT\_ID | MGR\_ID | JOB\_ID | SALARY |
| 101 | Smith | 20 | 120 | SA\_REP | 4000 |
| 102 | Martin | 10 | 105 | CLERK | 2500 |
| 103 | Chris | 20 | 120 | IT\_ADMIN | 4200 |
| 104 | John | 30 | 108 | HR\_CLERK | 2500 |
| 105 | Diana | 30 | 108 | HR\_MGR | 5000 |
| 106 | Bryan | 40 | 110 | AD\_ASST | 3000 |
| 108 | Jennifer | 30 | 110 | HR\_DIR | 6500 |
| 110 | Bob | 40 |  | EX\_DIR | 8000 |
| 120 | Ravi | 20 | 110 | SA\_DIR | 6500 |

Solution:

Select e.employee\_id “Emp\_id”, e.emp\_name “Employee”, e.salary, m.employee\_id “Mgr\_id”, m.emp\_name “Manager” FROM employees e, employee m

WHERE e.mgr\_id =m.employee\_id and e.salary> 4000

**Question 20**

Write program to reverse a linked list?

Solution: One method which I can think of is to reverse the list and then read it. But this involves changing the list which is bad.  
OR I can make a copy of the list and then reverse it, but this uses additional O(n) memory. Is there any better method which doesn't use extra memory and doesn't modify the list and runs in O(n) time

Void Reverse (Node head)  
{  
    Node prev= null;  
    Node current = head;  
    Node nextNode = null;  
  
        while (current!=null)  
        {  
                nextNode = current.Next;  
                current.Next = prev;  
                prev=current;  
                current = nextNode;       
  
        }  
        head = prev;  
  
}

Recursive Solution:

void ReadBackWard (Node n)  
{  
    if (n==null)  
        return;  
    else  
        ReadBackward(n.Next);  
  
    Console.WriteLine(n.Data);  
  
}