**Question 1**

How many Objects are eligible for garbage collection when the control reaches do Stuff

**public** **class** Test {

Integer story = **new** Integer(200);

Test go(Test cb) {

cb = **null**;

**return** cb;

}

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

Test c1 = **new** Test();

Test c2 = **new** Test();

Test c3 = c1.go(c2);

c1 = **null**;

// do Stuff

}

}

Solution: c1 and c3 and associated Integer objects it total of four objects will be garbage collected.

**Question 2**

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

Solution: The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesnt allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

**Question 3**

Explain the difference between the two statements in terms of memory allocation?

1. String s1 = new String(“abc”);

2. String s2 = “abc”;

Solution: Case 1: With new String even if the content is the same different objects are created and the address are assigned to different variables.

Case 2: With string with double quotes we can atmost create only one object in the entire program. If we create more objects the same address of the already created object will be assigned to the variable pointing to the newly created object.

**Question 4**

When do you get a concurrent modification exception when using an iterator?

Solution:   
*This exception may be thrown by methods that have detected concurrent modification of an object when such modification is not permissible.   
  
For example, it is not generally permssible for one thread to modify a Collection while another thread is iterating over it.   
  
Note that this exception does not always indicate that an object has been concurrently modified by a different thread. If a single thread issues a sequence of method invocations that violates the contract of an object, the object may throw this exception. For example, if a thread modifies a collection directly while it is iterating over the collection with a fail-fast iterator, the iterator will thow this exception.*

**Question 5**

Explain finalize method?

Solution: The "finalize" method is called when an object is removed from memory. If you need to do any cleanup (closing streams/files, displaying output, etc.) you can put it in an overridden finalize method.

**Question 6**

What is the difference between transient and volatile modifiers?

Solution:

Transient: The transient modifier applies to variable only

and it is not stored as part of it's objects persistent

state.Transient variables are not serialized.

Volatile: Volatile modifier applies to variables only and

it tells the compiler that the variable modified by

volatile can be changed unexpectedly by other parts of the

program.

**Question 7**

Why do we need wrapper classes? Explain with an example.

Solution: Wrapper classes allow primitive data types to be accessed as objects. They are one per primitive type: Boolean, Byte, Character, Double, Float, Integer, Long and Short. Wrapper classes make the primitive type data to act as objects.

Dealing with primitives as objects is easier at times. Most of the objects collection store objects and not primitive types. Many utility methods are provided by wrapper classes. To get these advantages we need to use wrapper classes. As they are objects, they can be stored in any of the collection and pass this collection as parameters to the methods.

Character wrapper class

**public class**IsDemo {  
  
  **public static void**main(String[] args) {  
  
    **char**a[] = {'a','b','5','?','A',' '};  
  
    **for**(**int**i=0;i<a.length;i++){  
  
      **if**(Character.isDigit(a[i]))  
        System.out.println(a[i] + "is a digit ");  
      **if**(Character.isLetter(a[i]))  
        System.out.println(a[i] + "is a letter ");  
      **if**(Character.isWhitespace(a[i]))  
        System.out.println(a[i] + "is a White Space ");  
      **if**(Character.isLowerCase(a[i]))  
        System.out.println(a[i] + "is a lower case ");  
      **if**(Character.isLowerCase(a[i]))  
        System.out.println(a[i] + "is a upper case ");  
    }  
  
  }  
}

**Question 8**

What is the difference between preemptive scheduling and time slicing?

Solution: Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

**Question 9**

Assuming that the serializeBanana() and deserializeBanana() methods will correctly use Java serialization what would be the output of the program.

**import** java.io.\*;

**class** Food **implements** Serializable {**int** good = 3; }

**class** Fruit **extends** Food { **int** juice = 5; }

**public** **class** Banana **extends** Fruit {

**int** yellow = 4;

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

Banana b = **new** Banana();

Banana b2;

b.serializeBanana(b);

b2 = b.deserializeBanana();

System.*out*.println("restore " + b2.yellow+ b2.juice + b2.good);

}

}

**Question 10**

Provide Output for the program

**public** **class** test **extends** Thread{

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

test t = **new** test();

Thread x = **new** Thread(t);

x.start();

}

**public** **void** run() {

**for** (**int** i = 0; i < 3; ++i) {

System.*out*.print(i + "..");

}

}

}

Solution:

0..1..2..

**Question 11**

Which method retrieves all cookies sent in HttpServletRequest request?

Solution: **Cookie[] cookies = request.getCookies();**

**Question 12**

What are implicit Objects? List them.

Solution:

Certain objects that are available for the use in JSP documents without being declared first. These objects are parsed by the JSP engine and inserted into the generated servlet. The implicit objects re listed below

|  |
| --- |
| * request * response * pageContext * session * application * out * config * page * exception |

**Question 13**

What is the best case, worst case running time of heapsort when all the elements are distinct?

Solution:

Best Case: O(n)

Worst case: O(n Log n)

**Question 14**

Write a program/pseudocode to print a histogram of the lengths of words in its input with vertical orientation.

Solution: Program in c

**#include <stdio.h>**

**#define MAXWORDLEN 10**

**int** main(**void**)

{

**int** c;

**int** inspace = **0**;

**long** lengtharr[MAXWORDLEN + **1**];

**int** wordlen = **0**;

**int** firstletter = **1**;

**long** thisval = **0**;

**long** maxval = **0**;

**int** thisidx = **0**;

**int** done = **0**;

**for**(thisidx = **0**; thisidx <= MAXWORDLEN; thisidx++)

{

lengtharr[thisidx] = **0**;

}

**while**(done == **0**)

{

c = getchar();

**if**(c == **' '** || c == **'\t'** || c == **'\n'** || c == EOF)

{

**if**(inspace == **0**)

{

firstletter = **0**;

inspace = **1**;

**if**(wordlen <= MAXWORDLEN)

{

**if**(wordlen > **0**)

{

thisval = ++lengtharr[wordlen - **1**];

**if**(thisval > maxval)

{

maxval = thisval;

}

}

}

**else**

{

thisval = ++lengtharr[MAXWORDLEN];

**if**(thisval > maxval)

{

maxval = thisval;

}

}

}

**if**(c == EOF)

{

done = **1**;

}

}

**else**

{

**if**(inspace == **1** || firstletter == **1**)

{

wordlen = **0**;

firstletter = **0**;

inspace = **0**;

}

++wordlen;

}

}

**for**(thisval = maxval; thisval > **0**; thisval--)

{

printf("%4d | ", thisval);

**for**(thisidx = **0**; thisidx <= MAXWORDLEN; thisidx++)

{

**if**(lengtharr[thisidx] >= thisval)

{

printf("\* ");

}

**else**

{

printf(" ");

}

}

printf("\n");

}

printf(" +");

**for**(thisidx = **0**; thisidx <= MAXWORDLEN; thisidx++)

{

printf("---");

}

printf("\n ");

**for**(thisidx = **0**; thisidx < MAXWORDLEN; thisidx++)

{

printf("%2d ", thisidx + **1**);

}

printf(">%d\n", MAXWORDLEN);

**return** **0**;

}

**Question 15**

Provide a pseudocode for efficiently finding loops in a singly linked list.

Solution:

The standard answer is to take two iterators at the beginning, increment the first one once, and the second one twice. Check to see if they point to the same object. Then repeat until the one that is incrementing twice either hits the first one or reaches the end.

This algorithm finds any circular link in the list, not just that it's a complete circle.

bool hasCircle(List l)  
{  
   Iterator i = l.begin(), j = l.begin();  
   while (true) {  
      // increment the iterators, if either is at the end, you're done, no circle  
      if (i.hasNext())  i = i.next(); else return false;  
  
      // second iterator is travelling twice as fast as first  
      if (j.hasNext())  j = j.next(); else return false;  
      if (j.hasNext())  j = j.next(); else return false;  
  
      // this should be whatever test shows that the two  
      // iterators are pointing at the same place  
      if (i.getObject() == j.getObject()) {   
          return true;  
      }   
   }  
}

**Question 16**

What method would you use to find the middle element in a linked list and why?

Solution: Best solution: Use two pointers and increment one variable by one and the second one by 2. When the second variable reaches the end of the list the first one is at the position of the middle element.

**Question 17**

Analyze the following data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

Write a query to get the list of ID, Name and salary of the employee, and the ID and name of the employee’s manager, for all the employees who have a manager and earn more than 4000.

Solution: 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 18**

What is the difference between SAX and DOM parser?

Solution: **SAX Parser:**

·        A SAX (**S**imple **A**PI for **X**ML) parser does not create any internal structure. Instead, it takes the occurrences of components of an input document **as events**, and tells the client what it reads as it reads through the input document.

·        A SAX parser serves the client application always only with pieces of the document at any given time.

·        A SAX parser, however, is much more space efficient in case of a big input document (because it creates no internal structure). What’s more, it runs faster and is easier to learn than DOM parser because its API is really simple. But from the functionality point of view, it provides a fewer functions, which means that the users themselves have to take care of more, such as creating their own data structures.

**DOM Parser:**

·        A DOM (Document Object Model) parser creates a tree structure in memory from an input document and then waits for requests from client.

·        A DOM parser always serves the client application with the entire document no matter how much is actually needed by the client.

·        A DOM parser is rich in functionality. It creates a DOM tree in memory and allows you to access any part of the document repeatedly and allows you to modify the DOM tree. But it is space inefficient when the document is huge, and it takes a little bit longer to learn how to work with it.

**Question 19**

Write an efficient program to split a comma separated string?

Solution:

string fruit = "Apple,Banana,Orange,Strawberry";

string[] split = fruit.Split(',');

foreach (string item in split)

{

Console.WriteLine(item);

}

**Question 20**

Explain Serialization and deserialization?

Solution: Serialization involves saving the current state of an object to a stream, and restoring an equivalent object from that stream. The stream functions as a container for the object. Its contents include a partial representation of the object's internal structure, including variable types, names, and values. The inverse process is deserialization.