Technical Questions

What will be the output of the program in 16-bit platform (under DOS)?

```
#include<stdio.h>

int main()
{
   struct node
   {
      int data;
      struct node *link;
   };
   struct node *p, *q;
   p = (struct node *) malloc(sizeof(struct node));
   q = (struct node *) malloc(sizeof(struct node));
   printf("%d, %d\n", sizeof(p), sizeof(q));
   return 0;
}
```

- A. 2, 2
- B. 8, 8
- C. 5, 5
- D. 4, 4

**Answer:** Option **A**

2. What will be the output of the program?

```
#include<stdio.h>
int X=40;
int main()
{
   int X=20;
   printf("%d\n", X);
   return 0;
}
```

- A. 20
- B. 40
- C. Error
- D. No Output

**Answer:** Option **A**

**Explanation:**

Whenever there is conflict between a local variable and global variable, the local variable gets priority.

3. What is the output of the program given below ?

```
#include<stdio.h>
int main()
{
   enum status { pass, fail, atkt};
   enum status stud1, stud2, stud3;
   stud1 = pass;
   stud2 = atkt;
   stud3 = fail;
   printf("%d, %d, %d\n", stud1, stud2, stud3);
   return 0;
}
```

A. 0, 1, 2

B. 1, 2, 3

C. 0, 2, 1

D. 1, 3, 2

**Answer:** Option **C**

**Explanation:**

enum takes the format like {0,1,2..) so pass=0, fail=1, atkt=2

stud1 = pass (value is 0)

stud2 = atkt (value is 2)

stud3 = fail (value is 1)

Hence it prints 0, 2, 1

---

4. What will be the output of the program (sample.c) given below if it is executed from the command line?
cmd> sample friday tuesday sunday

```
/* sample.c */
#include<stdio.h>

int main(int argc, char *argv[])
{
   printf("%c", **++argv);
   return 0;
}
```

A. s

B. f

C. sample

D. friday

---

**Answer:** Option **B**

5. The itoa function can convert an integer in decimal, octal or hexadecimal form to a string.

A. Yes B. No

**Answer:** Option **A**

**Explanation:**

itoa() takes the integer input value input and converts it to a number in base radix. The resulting number a sequence of base-radix digits.

**Example**:

```
/* itoa() example */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  int no;
  char buff [50];
  printf ("Enter number: ");
  scanf ("%d",&no);

  itoa (no,buff,10);
  printf ("Decimal: %s\n",buff);

  itoa (no,buff,2);
  printf ("Binary: %s\n",buff);

  itoa (no,buff,16);
  printf ("Hexadecimal: %s\n",buff);

  return 0;
}
```

**Output**:

```
Enter a number: 1250
Decimal: 1250
Binary: 10011100010
Hexadecimal: 4e2
```

6. Assume integer is 2 bytes wide. How many bytes will be allocated for the following code?

```
#include<stdio.h>
#include<stdlib.h>
#define MAXROW 3
#define MAXCOL 4
```

```
int main()
{
   int (*p)[MAXCOL];
   p = (int (*) [MAXCOL])malloc(MAXROW *sizeof(*p));
   return 0;
}
```

- A. 56 bytes
- B. 128 bytes
- C. 24 bytes
- D. 12 bytes

**Answer:** Option **C**

7. Point out the error in the program

```
#include<stdio.h>

int main()
{
   int i;
   #if A
      printf("Enter any number:");
      scanf("%d", &i);
   #elif B
      printf("The number is odd");
   return 0;
}
```

- A. Error: unexpected end of file because there is no matching #endif
- B. The number is odd
- C. Garbage values
- D. None of above

**Answer:** Option **A**

**Explanation:**

The conditional macro #if must have an #endif. In this program there is no #endifstatement written.

8. What will be the output of the program?

```
#include<stdio.h>

int main()
{
   unsigned int res;
   res = (64 >>(2+1-2)) & (~(1<<2));
   printf("%d\n", res);
   return 0;
}
```

A. 32

B. 64

C. 0

D. 128

**Answer:** Option **A**

9. Point out the error in the program?

```
#include<stdio.h>

int main()
{
   struct a
   {
      float category:5;
      char scheme:4;
   };
   printf("size=%d", sizeof(struct a));
   return 0;
}
```

A. Error: invalid structure member in printf

B. Error in this float category:5; statement

C. No error

D. None of above

**Answer:** Option **B**

**Explanation:**

Bit field type must be signed int or unsigned int.

The char type: char scheme:4; is also a valid statement.

---

10. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
   printf("%%%%%\n");
   return 0;
}
```

A. %%%%%

B. %%

C. No output

D. Error

**Answer:** Option **B**

---

11. Are the three declarations char **apple, char *apple[], and char apple[][]same?

A. True

B. False

**Answer:** Option **B**

---

12. What will be the output of the program (Turbo C in 16 bit platform DOS) ?

```
#include<stdio.h>
#include<string.h>

int main()
{
   char *str1 = "India";
   char *str2 = "BIX";
   char *str3;
   str3 = strcat(str1, str2);
   printf("%s %s\n", str3, str1);
   return 0;
}
```

A. IndiaBIX India

B. IndiaBIX IndiaBIX

C. India India

D. Error

---

**Answer:** Option **B**

**Explanation:**

It prints 'IndiaBIX IndiaBIX' in TurboC (in 16 bit platform).

It may cause a 'segmentation fault error' in GCC (32 bit platform).

---

13. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
   char str[25] = "IndiaBIX";
   printf("%s\n", &str+2);
   return 0;
}
```

A. Garbage value

B. Error

C. No output

D. diaBIX

**Answer:** Option **A**

**Explanation:**

**Step 1**: char str[25] = "IndiaBIX"; The variable str is declared as an array of characteres and initialized with a string "IndiaBIX".

**Step 2**: printf("%s\n", &str+2);

=> In the printf statement %s is string format specifier tells the compiler to print the string in the memory of &str+2

=> &str is a location of string "IndiaBIX". Therefore &str+2 is another memory location.

Hence it prints the Garbage value.

---

14. Which standard library function will you use to find the last occurance of a character in a string in C?

A. strnchar()

B. strchar()

C. strrchar()

D. strrchr()

**Answer:** Option **D**

---

**Explanation:**

strrchr() returns a pointer to the last occurrence of character in a string.

**Example**:

```
#include <stdio.h>
#include <string.h>

int main()
{
   char str[30] = "12345678910111213";
   printf("The last position of '2' is %d.\n",
          strrchr(str, '2') - str);
   return 0;
}
```

**Output**: The last position of '2' is 14.

15. How will you free the memory allocated by the following program?

```
#include<stdio.h>
#include<stdlib.h>
#define MAXROW 3
#define MAXCOL 4

int main()
{
   int **p, i, j;
   p = (int **) malloc(MAXROW * sizeof(int*));
   return 0;
}
```

- A. memfree(int p);
- B. dealloc(p);
- C. malloc(p, 0);
- D. free(p);

**Answer:** Option **D**

16. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
   static int arr[] = {0, 1, 2, 3, 4};
   int *p[] = {arr, arr+1, arr+2, arr+3, arr+4};
   int **ptr=p;
   ptr++;
   printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
```

```
    *ptr++;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    *++ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    ++*ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    return 0;
}
```

- A.
  0, 0, 0
  1, 1, 1
  2, 2, 2
  3, 3, 3
- B.
  1, 1, 2
  2, 2, 3
  3, 3, 4
  4, 4, 1
- C.
  1, 1, 1
  2, 2, 2
  3, 3, 3
  3, 4, 4
- D.
  0, 1, 2
  1, 2, 3
  2, 3, 4
  3, 4, 5

**Answer:** Option **C**

---

17. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n", sizeof(arr)/sizeof(arr[0]));
    return 0;
}
```

- A. 5
- B. 4
- C. 6
- D. 7

**Answer:** Option **B**

**Explanation:**

The sizeof function return the given variable. Example: float a=10; sizeof(a) is 4 bytes

**Step 1**: float arr[] = {12.4, 2.3, 4.5, 6.7}; The variable arr is declared as an floating point array and it is initialized with the values.

**Step 2**: printf("%d\n", sizeof(arr)/sizeof(arr[0]));

The variable arr has 4 elements. The size of the float variable is 4 bytes.

Hence 4 elements x 4 bytes = 16 bytes

---

sizeof(arr[0]) is 4 bytes

Hence 16/4 is 4 bytes

Hence the output of the program is '4'.

---

18. What will be the output of the program?

```
#include<stdio.h>
int main()
{
   float a = 0.7;
   if(0.7 > a)
      printf("Hi\n");
   else
      printf("Hello\n");
   return 0;
}
```

- A. Hi
- B. Hello
- C. Hi Hello
- D. None of above

**Answer:** Option **A**

**Explanation:**

if(0.7 > a) here a is a float variable and 0.7 is a double constant. The double constant 0.7 is greater than the float variable a. Hence the if condition is satisfied and it prints 'Hi'
**Example**:

```
#include<stdio.h>
int main()
{
   float a=0.7;
   printf("%.10f %.10f\n",0.7, a);
   return 0;
}
```

**Output**:
0.7000000000 0.6999999881

---

19. What will be the output of the program?

```
#include<stdio.h>

int main()
{
   int i;
   char c;
```

```
    for(i=1; i<=5; i++)
    {
        scanf("%c", &c); /* given input is 'b' */
        ungetc(c, stdout);
        printf("%c", c);
        ungetc(c, stdin);
    }
    return 0;
}
```

A. bbbb

B. bbbbb

C. b

D. Error in ungetc statement.

**Answer:** Option **C**

**Explanation:**

The ungetc() function pushes the character c back onto the named input stream, which must be open for reading.

This character will be returned on the next call to getc or fread for that stream.

One character can be pushed back in all situations.

A second call to ungetc without a call to getc will force the previous character to be forgotten.

---

20. What will be the output of the program?

```
#include<stdio.h>
void fun(int*, int*);
int main()
{
    int i=5, j=2;
    fun(&i, &j);
    printf("%d, %d", i, j);
    return 0;
}
void fun(int *i, int *j)
{
    *i = *i**i;
    *j = *j**j;
}
```

A. 5, 2

B. 10, 4

C. 2, 5

D. 25, 4

**Answer:** Option **D**

---

**Explanation:**

**Step 1**: int i=5, j=2; Here variable i and j are declared as an integer type and initialized to 5 and 2 respectively.

**Step 2**: fun(&i, &j); Here the function fun() is called with two parameters &i and &j(The & denotes call by reference. So the address of the variable i and j are passed. )

**Step 3**: void fun(int *i, int *j) This function is called by reference, so we have to use * before the parameters.

**Step 4**: *i = *i**i; Here *i denotes the value of the variable i. We are multiplying5*5 and storing the result 25 in same variable i.

**Step 5**: *j = *j**j; Here *j denotes the value of the variable j. We are multiplying2*2 and storing the result 4 in same variable j.

**Step 6**: Then the function void fun(int *i, int *j) return back the control back tomain() function.

**Step 7**: printf("%d, %d", i, j); It prints the value of variable i and j.

Hence the output is 25, 4.

---

21.Is it possible for a member function of a class to activate another member function of the same class?
a. no
b. yes,but only public member functions.
c. Yes, only private member functions
d. Yes, both public & private member functions can be activated

22. Can 2 classes contain member functions with the same name?
a. no
b. yes, but only if the 2 classes have the same name
c. yes, but only if the main program does not declare both kinds
d. yes, this is always allowed

23.what is the purpose of template functions?
a. to allow a single function to be used with varying types of arguments
b. to hide the same name of the function from the linker
c. to implement container classes
d. to permit the use of the debugger without the –gstabs flag

24.what kind of functions can access private member variables of a class
a. friend function of a class
b. private member functions of the class
c. public member functions of the class
e. all of the above
f. none

25. what technique is used to step thro' items of a container class?
a. copy constructor
b. default constructor
c. iterator (ans)

26. what is the term used to describe the situation when a derived class provides a function already provided in the base class ?
a. inheritning
b. overriding

c. abstraction
d. none

27. what r the characteristics of object, in general
a. it has name, properties
b. name, properties, behaviour
c. it can act on receiving a message
d. a &c
e. b&c

28. bundling data members together with the functions is called
a. data hiding
b. encapsulation (ans)
c. data morphism
d. data binding

29.why is OO programming useful?
a. It supports good s/w engg. Practices
b. It promotes thinking abt s/w in a way that models the way we think & interact naturally
c. Supports code reuse
d. A &b
e. A,b & c

30. suppose tu want 2 develop 2 diff, + operators that calculate & return an object of the same class. Which stmt is correct?
a. 1 operator must be a frnd & the other must not
b. 1 operator must be public & the other private
c. operators must have different parameter lists
d. it is not possible 2 have 2 different + operators

31.what is inheritance?
a. a relationship in which the structure & functionality of a class (“child ) is defined in terms of the structure & functionality of another(parent)
b. a relationship in which the structure & functionality of a class (“parent” ) is defined in terms of the structure & functionality of another(“child”)
c. a relationship in which the structure & functionality of a class make calls 2 functionalty of another
d. a relationship in which the structure of a class includes 1 or more instances of another
e. all of above

32. what is relationship betn . a class & its public parent class
a. “.is a….”
b. “has a”
c. “is implemented as a ……”
d. uses a
e. becomes a

33. how does inheritance relate 2 abstraction?
A . a base class is an abstraction of all its derived classes
b. a derived class is an abstraction of all its base classes
d. Base & derived classes are abstractions for each other
e. Inheritance prevents abstration
f. No relationship betn them

34. why can’t compiler “hard-code” direct calls to virtual functions?
a. b’coz hard –coding the fn. Call wud slow compilation of the program too much
b. b’coz hard –coding the fn. Call wud slow execution of the program too much
c. b’coz the correct fn. To call generally isn’t know at compile –time
d. b’coz virtual fns are always at compile-time so no run-time code is needed at all.
e. Compiler does hard-code direct calls 2 virtual functions.

35. what is accessibility of a protected member of a base class which is inherited privately?

a. private: b'coz the private inheritance makes everything frm the base class private in the derived class (ans)

36. Indexing must be applied on fields that are
a. seldom refernced in query
b. containing few unique columns
c. on frequently searched columns
d. changed frequently

37. The relational data model includes several types of data integrity .which type of data integrity is ensured by the use of a primary key?
a. entity integrity
b. domain integrity
c. referential integrity
d. relational integrity

38. a student has 3 diff. Teachers . Each of these teachers has several students. Which type ofrelationship in relation model would effective represent this?
a. 3 one-to-many relationships
b. two many-to-one relationships
c. a single many-to –many relationshjips
d. at least 2 many-to-many relationships

39. which of the following properties are required of a column or column sets for the column to function as a primary key?
a. no nulls
b. a clustered index
c. a non-clustered index
d. ther must be atleast 1 foreign key on tha same table.

40. select a.pk, fn, b.zip, c.city from a, b, c wher a.pk=b.pk
tables a,b,&c each contain 100 rows & the primary key for tables a & b is pk. What is the maxm. No. of rows that could be returned by query
a. 100
b. 0
c. 100,000
d. 10,000
e. 1,000,000

41. which of the following is only used with a foreign key constraint ?
a . primary key constraint
b. the refereces clause
c. a check constraint

42. which is used to return the no. of rows in a table?
a. sel;ect all count from tablename
b. select count(*) from tablename
c. select numrows from table name

43. when writing sql queries , how do u control the query execution plan used by the databases
a. query directly against an idex instead of table
b. define a static cursor & specify the "using specific index' keyword
c. most databases provide propreitory methods to give database hint
d. enclose quey in atransaction & add a statement using the "set search" keyword

44. Which is not allowed in a trigger
a. update
b. select into
c. while
d. begin

45.select type, avg(price), min(price) from product group by category
what is wrong??
a. u can't include multiple aggregates in a select list
b. there is no where clause
c. nothing

46. What are the common mechanisms used for session tracking?

Cookies SSL sessions URL- rewriting

47. Explain Servlet Context.

ServletContext interface is a window for a servlet to view it's environment. A servlet can use this interface to get information such as initialization parameters for the web application or servlet container's version. Every web application has one and only one ServletContext and is accessible to all active resource of that application.

48. What is preinitialization of a servlet?

A container doesnot initialize the servlets ass soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

49. What is the difference between Difference between doGet() and doPost()?

A doGet() method is limited with 2k of data to be sent, and doPost() method doesn't have this limitation. A request string for doGet() looks like the following: http://www.allapplabs.com/svt1?p1=v1&p2=v2&…&pN=vN

doPost() method call doesn't need a long text tail after a servlet name in a request. All parameters are stored in a request itself, not in a request string, and it's impossible to guess the data transmitted to a servlet only looking at a request string.

50. What is the difference between HttpServlet and GenericServlet?

A GenericServlet has a service() method aimed to handle requests. HttpServlet extends GenericServlet and adds support for doGet(), doPost(), doHead() methods (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() methods (HTTP 1.1). Both these classes are abstract.

51. What is a output comment?
A comment that is sent to the client in the viewable page source.The JSP engine handles an output comment as uninterpreted HTML text, returning the comment in the HTML output sent to the client. You can see the comment by viewing the page source from your Web browser. JSP Syntax Example 1 Displays in the page source:

52. What is a Hidden Comment?
A comments that documents the JSP page but is not sent to the client. The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags. A hidden comment is not sent to the client, either in the displayed JSP page or the HTML page source. The hidden comment is useful when you want to hide or "comment out" part of your JSP page. You can use any characters in the body

of the comment except the closing –%> combination. If you need to use –%> in your comment, you can escape it by typing –%\>. JSP Syntax Examples

53. What is a _Expression?
An _expression tag contains a scripting language _expression that is evaluated, converted to a String, and inserted where the _expression appears in the JSP file. Because the value of an _expression is converted to a String, you can use an _expression within text in a JSP file. Like You cannot use a semicolon to end an _expression

54. What is a Declaration?
A declaration declares one or more variables or methods for use later in the JSP source file. A declaration must contain at least one complete declarative statement. You can declare any number of variables or methods within one declaration tag, as long as they are separated by semicolons. The declaration must be valid in the scripting language used in the JSP file.

55. Does importing a package imports the subpackages as well? e.g. Does importing com.MyTest.* also import com.MyTest.UnitTests.*? No you will have to import the subpackages explicitly. Importing com.MyTest.* will import classes in the package MyTest only. It will not import any class in any of it's subpackage.

56. What is a Scriptlet?
A scriptlet can contain any number of language statements, variable or method declarations, or expressions that are valid in the page scripting language.Within scriptlet tags, you can

1.Declare variables or methods to use later in the file (see also Declaration).

2.Write expressions valid in the page scripting language (see also _Expression).

3.Use any of the JSP implicit objects or any object declared with a tag.

You must write plain text, HTML-encoded text, or other JSP tags outside the scriptlet. Scriptlets are executed at request time, when the JSP engine processes the client request. If the scriptlet produces output, the output is stored in the out object, from which you can display it.

57. What are implicit objects? List them?
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

58. Difference between forward and send Redirect?
When you invoke a forward request, the request is sent to another resource on the server, without the client being informed that a different resource is going to process the request. This process occurs completly with in the web container. When a sendRedirtect method is invoked, it causes the web container to return to the browser indicating that a new URL should be requested. Because the browser issues a completly new request any object that are stored as request attributes before the redirect occurs will be lost. This extra round trip a redirect is slower than forward.

59. What are the different scope valiues for the ?
The different scope values for are
1. page
2. request
3.session
4.application

60. Explain the life-cycle mehtods in JSP?
THe generated servlet class for a JSP page implements the HttpJspPage interface of the javax.servlet.jsp package. Hte HttpJspPage interface extends the JspPage interface which inturn extends the Servlet

interface of the javax.servlet package. the generated servlet class thus implements all the methods of the these three interfaces.

The JspPage interface declares only two mehtods – jspInit() and jspDestroy() that must be implemented by all JSP pages regardless of the client-server protocol.

However the JSP specification has provided the HttpJspPage interfaec specifically for the JSp pages serving HTTP requests.
This interface declares one method _jspService(). The jspInit()- The container calls the jspInit() to initialize te servlet instance.It is called before any other method, and is called only once for a servlet instance. The _jspservice()- The container calls the _jspservice() for each request, passing it the request and the response objects.
The jspDestroy()- The container calls this when it decides take the instance out of service. It is the last method called n the servlet instance.

61.What is the common usage of serialization?
Whenever an object is to be sent over the network, objects need to be serialized. Moreover if the state of an object is to be saved, objects need to be serilazed.

62. What is Externalizable interface?
Externalizable is an interface which contains two methods readExternal and writeExternal. These methods give you a control over the serialization mechanism. Thus if your class implements this interface, you can customize the serialization process by implementing these methods.

63. What happens to the object references included in the object?
The serialization mechanism generates an object graph for serialization. Thus it determines whether the included object references are serializable or not. This is a recursive process. Thus when an object is serialized, all the included objects are also serialized alongwith the original obect.

64. What one should take care of while serializing the object?
One should make sure that all the included objects are also serializable. If any of the objects is not serializable then it throws a NotSerializableException.

65. What if the main method is declared as private?
The program compiles properly but at runtime it will give “Main method not public.” message.

66. What if the static modifier is removed from the signature of the main method?
Program compiles. But at runtime throws an error “NoSuchMethodError”.

67. What if I write static public void instead of public static void?
Program compiles and runs properly.

68. What if I do not provide the String array as the argument to the method?
Program compiles but throws a runtime error “NoSuchMethodError”.

69. What is the first argument of the String array in main method?
The String array is empty. It does not have any element. This is unlike C/C++ where the first element by default is the program name.

70. If I do not provide any arguments on the command line, then the String array of Main method will be empty of null?
It is empty. But not null.

71. How can one prove that the array is not null but empty?
Print args.length. It will print 0. That means it is empty. But if it would have been null then it would have thrown a NullPointerException on attempting to print args.length.

72. What environment variables do I need to set on my machine in order to be able to run Java programs?
CLASSPATH and PATH are the two variables.

73. Can an application have multiple classes having main method?
Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned. Hence there is not conflict amongst the multiple classes having main method.

74. Can I have multiple main methods in the same class?
No the program fails to compile. The compiler says that the main method is already defined in the class.

75. Do I need to import java.lang package any time? Why ?
No. It is by default loaded internally by the JVM.

76. Can I import same package/class twice? Will the JVM load the package twice at runtime?
One can import the same package or same class multiple times. Neither compiler nor JVM complains abt it. And the JVM will internally load the class only once no matter how many times you import the same class.

77. What are Checked and UnChecked Exception?
A **checked exception** is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses. Making an exception checked forces client programmers to deal with the possibility that the exception will be thrown. eg, IOException thrown by java.io.FileInputStream's read() method

**checked exceptions** are RuntimeException and any of its subclasses. Class Error and its subclasses also are unchecked. With an unchecked exception, however, the compiler doesn't force client programmers either to catch the exception or declare it in a throws clause. In fact, client programmers may not even know that the exception could be thrown. eg, StringIndexOutOfBoundsException thrown by String's charAt() method Checked exceptions must be caught at compile time. Runtime exceptions do not need to be. Errors often cannot be.

78. What is Overriding?
When a class defines a method using the same name, return type, and arguments as a method in its superclass, the method in the class overrides the method in the superclass. When the method is invoked for an object of the class, it is the new definition of the method that is called, and not the method definition from superclass. Methods may be overridden to be more public, not more private.

79. What are different types of inner classes?
They are Nested -level classes, Member classes, Local classes, Anonymous classes

**Nested -level classes**- If you declare a class within a class and specify the static modifier, the compiler treats the class just like any other -level class. Any class outside the declaring class accesses the nested class with the declaring class name acting similarly to a package. eg, outer.inner. -level inner classes implicitly have access only to static variables.There can also be inner interfaces. All of these are of the nested -level variety.

**Member classes** – Member inner classes are just like other member methods and member variables and access to the member class is restricted, just like methods and variables. This means a public member class acts similarly to a nested -level class. The primary difference between member classes and nested -level classes is that member classes have access to the specific instance of the enclosing class.

**Local classes** – Local classes are like local variables, specific to a block of code. Their visibility is only within the block of their declaration. In order for the class to be useful beyond the declaration block, it would need to implement a more publicly available interface.Because local classes are not members, the modifiers public, protected, private, and static are not usable.

**Anonymous classes** – Anonymous inner classes extend local inner classes one level further. As anonymous classes have no name, you cannot provide a constructor.

80. Are the imports checked for validity at compile time?

e.g. will the code containing an import such as java.lang.ABCD compile? Yes the imports are checked for the semantic validity at compile time. The code containing above line of import will not compile. It will throw an error saying,can not resolve symbol symbol : class ABCD location: package io import java.io.ABCD;

81. Does importing a package imports the subpackages as well?

e.g. Does importing com.MyTest.* also import com.MyTest.UnitTests.*? No you will have to import the subpackages explicitly. Importing com.MyTest.* will import classes in the package MyTest only. It will not import any class in any of it's subpackage.

82. What is the difference between declaring a variable and defining a variable?

In declaration we just mention the type of the variable and it's name. We do not initialize it. But defining means declaration + initialization. e.g String s; is just a declaration while String s = new String ("abcd"); Or String s = "abcd"; are both definitions.

83. What is the default value of an object reference declared as an instance variable?

null unless we define it explicitly.

84. Can a level class be private or protected? No.

A level class can not be private or protected. It can have either "public" or no modifier. If it does not have a modifier it is supposed to have a default access. If a level class is declared as private the compiler will complain that the "modifier private is not allowed here". This means that a level class can not be private. Same is the case with protected.

85. What type of parameter passing does Java support? In Java the arguments are always passed by value .

86. Primitive data types are passed by reference or pass by value?

Primitive data types are passed by value.

87. Objects are passed by value or by reference?

Java only supports pass by value. With objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same object .

88. What is serialization?

Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

89. How do I serialize an object to a file?

The class whose instances are to be serialized should implement an interface Serializable. Then you pass the instance to the ObjectOutputStream which is connected to a fileoutputstream. This will save the object to a file.

90. Which methods of Serializable interface should I implement?

The serializable interface is an empty interface, it does not contain any methods. So we do not implement any methods.

91. How can I customize the seralization process?

i.e. how can one have a control over the serialization process? Yes it is possible to have control over serialization process. The class should implement Externalizable interface. This interface contains two methods namely readExternal and writeExternal. You should implement these methods and write the logic for customizing the serialization process.

92. What is the common usage of serialization?

Whenever an object is to be sent over the network, objects need to be serialized. Moreover if the state of an object is to be saved, objects need to be serilazed.

93. What is Externalizable interface?

Externalizable is an interface which contains two methods readExternal and writeExternal. These methods give you a control over the serialization mechanism. Thus if your class implements this interface, you can customize the serialization process by implementing these methods.

94. What happens to the object references included in the object?

The serialization mechanism generates an object graph for serialization. Thus it determines whether the included object references are serializable or not. This is a recursive process. Thus when an object is serialized, all the included objects are also serialized alongwith the original obect.

95. What one should take care of while serializing the object?

One should make sure that all the included objects are also serializable. If any of the objects is not serializable then it throws a NotSerializableException.

96. What happens to the static fields of a class during serialization?

Are these fields serialized as a part of each serialized object? Yes the static fields do get serialized. If the static field is an object then it must have implemented Serializable interface. The static fields are serialized as a part of every object. But the commonness of the static fields across all the instances is maintained even after serialization.

97. How are Observer and Observable used?

Objects that subclass the Observable class maintain a list of observers. When an Observable object is updated it invokes the update() method of each of its observers to notify the observers that it has changed state. The Observer interface is implemented by objects that observe Observable objects. [Received from Venkateswara Manam]

98. What is synchronization and why is it important?

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object’s value. This often leads to significant errors.

99. How does Java handle integer overflows and underflows?

It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

100. Does garbage collection guarantee that a program will not run out of memory?

Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to garbage collection .

101. What is the difference between preemptive scheduling and time slicing?

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.

102. When a thread is created and started, what is its initial state?

A thread is in the ready state after it has been created and started.

103. What is the purpose of finalization?

The purpose of finalization is to give an unreachable object the opportunity to perform any cleanup processing before the object is garbage collected.

104. What is the Locale class?

The Locale class is used to tailor program output to the conventions of a particular geographic, political, or cultural region.

105. What is the difference between a while statement and a do statement?

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute the body of a loop at least once.

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

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

107. How are this() and super() used with constructors?

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

108. Explain the life cycle methods of a Servlet.
The javax.servlet.Servlet interface defines the three methods known as life-cycle method. public void init(ServletConfig config) throws ServletException public void service( ServletRequest req, ServletResponse res) throws ServletException, IOException public void destroy() First the servlet is constructed, then initialized wih the init() method. Any request from client are handled initially by the service() method before delegating to the doXxx() methods in the case of HttpServlet. The servlet is removed from service, destroyed with the destroy() methid, then garbaged collected and finalized.

109. What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?

The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a “/” it is interpreted as relative to the current context root. The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must sart with a “/” and are interpreted as relative to curent context root.

110. Explain the directory structure of a web application.

The directory structure of a web application consists of two parts. A private directory called WEB-INF A public resource directory which contains public resource folder. WEB-INF folder consists of 1. web.xml 2. classes directory 3. lib directory