CLASS XII
GUESS PAPER
INFORMATICS PRAXCTICES

Java Objective Questions and Answer

Q1.Which of the following are compiling without warning or error?
a) float f=1.3;
b) char c="a";
c) byte b=257;
d) boolean b=null;
e) int i=10;

Q2.Which of the following are legal statements?
a) float f=1/3;
b) int i=1/3;
c) float f=1.01;
d) double d=999d;.

Q3.Which of the following are Java keywords?
a) NULL.
b) new.
c) instanceOf.
d) wend.

Q4.Which of the following are valid statements?
a) System.out.println(5+4);
b) int i=2+'3';
c) String s="one"+'Two';
d) byte b=255;

Q 5.Which of the following statements are true?
a) The garbage collection algorithm in Java is vendor implemented.
b) The size of primitives is platform dependent.
c) The default type for a numerical literal with decimal component is a float.
d) You can modify the value in an Instance of the Integer class with the setValue method.

Q6.Which of the following are true statements?
a) The creation of a named instance of the File class creates a matching file in the underlying operating system only when the close method is called.
b) The RandomAccessFile class allows you to move directly to any point a file.
c). I/O in Java can only be performed using the Listener classes.
d) The characteristics of an instance of the File class such as the directory separator, depend on the current underlying operating system.

Q 7.Which of the following statements are true?
a) The instanceof operator can be used to determine if a reference is an instance of a class, but not an interface.
b) The instanceof operator can be used to determine if a reference is an instance of a particular primitive wrapper class.
c) The instanceof operator will only determine if a reference is an instance of a class immediately above in the hierarchy but no further up the inheritance chain.
d) The instanceof operator can be used to determine if one reference is of the same class as another reference thus.

Q8.Which of the following statements are true?
a) An interface can only contain method and not variables.
b) Interfaces cannot have constructors.
c) A class may extend only one other class and implement only one interface.
d) Interfaces are the Java approach to addressing its lack of multiple inheritance, but require implementing classes to create the functionality of the Interfaces.

Q9.Which of the following are valid statements?
a) public class MyCalc extends Math
b) Math.max(5);
c) Math.round(9.99,1);
d)Math.mod(4,10);

Q10.Which of the following are methods of the Runnable interface?
a) run
b) start
c) yield
d) stop

Q11.Which of the following statements are true?
a) A byte can represent between -128 to 127.
b) A byte can represent between -127 to 128.
c) A byte can represent between -256 to 256.
d) A char can represent between -2x2 pow 16 2 x2 pow 16 - 1.

Q12.What will happen when you attempt to compile and run the following code?

```
class Base{
public void Base(){
System.out.println("Base");
}
}
public class Inner extends Base{
public static void main(String argv[]){
Inner i=new Inner();
}
}
```

a) Compile time error Base is a keyword.
b) Compilation and no output at runtime.
c) Output of Base.
d) Runtime error Base has no valid constructor.

Q13.You have a public class called myclass with the main method defined as follows
public static void main(String parm[]) {
System.out.println(parm[0]);
}
If you attempt to compile the class and run the program as follows
"java myclass hello"
What will happen?
a) Compile time error, main is not correctly defined.
b) Run time error, main is not correctly defined.
c) Compilation and output of java.
d) Compilation and output of hello.

Q14.Which of the following statements are true?
a) If a class has any abstract methods it must be declared abstract itself.
b) All methods in an abstract class must be declared as abstract.
c) When applied to a class, the final modifier means it cannot be sub-classed.
d) transient and volatile are Java modifiers.

Q15.Which of the following are valid methods?
a) public static native void amethod(){}
b) public static void amethod(){}
c) private protected void amethod(){}
d) static native void amethod();

Q16.Which of the following statements are true?
a) Constructors cannot have a visibility modifier.
b) Constructors can be marked public and protected, but not private.
c) Constructors can only have a primitive return type.
d) Constructors are not inherited.

Q17.What will happen when you attempt to compile and run the following class?
class Base{
Base(int i){
System.out.println("Base");
}
}
class Inner extends Base{
public static void main(String argv[]){
Inner s = new Inner();
}
void Inner (){
System.out.println("Inner");
}
}
a) Compilation and output of the string "Inner "at runtime.
b) Compile time error.
c) Compilation and no output at runtime.
d) Compilation and output of the string "Base".

Q18.Which of the following statements are true?

a) A static methods do not have access to the implicit variable called this.
b) A static method may be called without creating an instance of its class.
c) A static method may not be overridden to be non-static.
d) A static method may not be overloaded.

Q19.Which of the following will compile without error?
a)char c='1';
System.out.println(c>>1);
b)Integer i=Integer("1");
System.out.println(i>>1);
c)int i=1;
System.out.println(i<<<1);
d)int i=1;
System.out.println(i<<1);

Q 20.Which of the following are Java keywords?
a) sizeof
b) main
c) transient
d) volatile

Q21.Which of the following statements are true?
a) The default constructor has a return type of void.
b) The default constructor takes a parameter of void.
c) The default constructor takes no parameters.
d) The default constructor is not created if the class has any constructors of its own.

Q22.Which of the following statements are true?
a) All of the variables in an interface are implicitly static.
b) All of the variables in an interface are implicitly final.
c) All of the methods in an interface are implicitly abstract.
d) A method in an interface can access class level variables.

Q 23.Which of the following statements are true?
a) The String class is implemented as a char array, elements are addressed using the stringname[] convention.
b) The + operator is overloaded for concatenation for the String class.
c) Strings are a primitive type in Java and the StringBuffer is used as the matching wrapper type.
d) The size of a string can be retrieved using the length property.

Q 24.Which of the following statements are true?
a) A method in an interface must not have a body.
b) A class may extend one other class plus at most one interface.
c) A class may extends at most one other class plus implement many interfaces.
d) An class accesses an interface via the keyword uses.

Q 25.Which of the following statements are true?
a) The following statement will produce a result of 1.
System.out.println(-1 >>>2);
b) Performing an unsigned left shift (<<<) on a negative number will always produce a negative

number result.
c) The following statement will produce a result of zero,
System.out.println(1 >>1);
d) All the Java integral types are signed numbers

Declarations and **Access Control**

Questions 1. Which of these array declarations and instantiations are not legal?
Select all valid answers:
(a) int []a[] = new int[4][4];
(b) int a[][] = new int[4][4];
(c) int a[][] = new int[][4];
(d) int []a[] = new int[4][];
(e) int[][] a = new int[4][4];
(f) int [ ] [ ] a = new int[4][4];

Questions 2 Which of these array declarations and initialization are not legal?
Select all valid answers:
(a) int []i[] = {{1,2}, {1}, {}, {1,2,3}};
(b) int i[]= new int[2]{1, 2};
(c) int i[][] = new int[][]{{1,2,3}, {4,5,6}};
(d) int i[][] = {{1, 2}, new int[2]};
(e) int i[4] = {1, 2, 3, 4};

Questions 3 Given the following code, which statements can be placed at the indicated position without causing compilation errors?

```
public class ThisUsage {
int planets;
static int suns;

public void gaze() {
int i;
// … insert statements here …
}
}
```

Select all valid answers:
(a) i = this.planets;
(b) i = this.suns;
(c) this = new ThisUsage();
(d) this.i = 4;
(e) this.suns = planets;

Questions 4 Given the following pairs of method declarations, which of these statements ar true?

```
void fly(int distance) {}
int fly(int time, int speed) {return time*speed}
```

void fall(int time) {}
int fall(int distance) {return distance}

void glide(int time) {}
void Glide(int time) {}

Select all valid answers:
(a) The first pair of methods will compile correctly and overload the method name fly.
(b) The second pair of methods will compile correctly and overload the method name fall.
(c) The third pair of methods will compile correctly and overload the method name glide.
(d) The second pair of methods will not compile correctly.
(e) The third pair of methods will not compile correctly.

Questions 5 Given a class named Book, which of these would be valid definitions of constructors for the class? [1]
Select all valid answers:
(a) Book(Book b) {}
(b) Book Book() {}
(c) private final Book() {}
(d) void Book() {}
(e) public static void Book(String args[]) {}
(f) abstract Book() {}

Questions 6 Which of these statements are true?
Select all valid answers:
(a) All classes must define a constructor.
(b) A constructor can be declared private.
(c) A constructor can declare a return value.
(d) A constructor must initialize all the member variables of a class.
(e) A constructor can access the non-static members of a class.

Questions 7 What will be the result of attempting to compile the following program?

```
Public class MyClass {
Long var;

Public void MyClass(long param) { var = param; } // (1)

Public static void main( String args[] ) {
MyClass a, b;
A = new MyClass(); // (2)
B = new MyClass(5); // (3)
}
}
```

Select the one right answer:
(a) A compilation error will be encountered at (1), since constructors should not specify a return value.
(b) A compilation error will be encountered at (2), since the class does not have a default constructor
(c) A compilation error will be encountered at (3), since the class does not have a constructor accepting a single argument of type int.
(d) The program will compile correctly.

Questions 8 Given the following class, which of these are valid ways of referring to the class from outside of the package net.basemaster?
package net.basemaster;

public class Base {
// …
}

Select all valid answers:
(a) By simply referring to the class as Base.
(b) By simply referring to the class as basemaster.Base
(c) By simply referring to the class as net.basemaster.Base
(d) By importing net.basemaster.* and referring to the class as Base
(e) By importing the package net.* and referring to the class as basemaster.Base

Questions 9 Which one of the following class definitions is a legal definition of a class that cannot be instantiated?
Select the one right answer:
(a) class Ghost {
abstract void haunt()
}
(b) abstract class Ghost {
void haunt();
}
(c) abstract class Ghost {
void haunt() {};
}
(d) abstract Ghost {
abstract void haunt();
}
(e) static class Ghost {
abstract haunt();
}

Questions 10 Which of these statements concerning the use of modifiers are true?
Select all valid answers:
(a) If no accessibility modifier (public, protected and private) is given in a member declaration of a class, the member is only accessible for classes in the same package and subclasses of the class.
(b) You cannot specify visibility of local variables. They are always only accessible within the block in which they ardeclared.
(c) Subclasses of a class must reside in the same package as the class they extend.
(d) Local variables can be declared static.
(e) Objects themselves don not have visibility, only references to the object.

Questions 11 Given the following source code, which one of the lines that are commeneted out may be reinserted without introducing errors?
abstract class MyClass {
abstract void f();
final void g() {}
// final void h() {} // (1)

```
protected static int i;
private int j;
}
final class MyOtherClass extends MyClass {
// MyOtherClass(int n) { m = n; } // (2)

public static void main(String args[]) {
MyClass mc = new MyOtherClass();
}

void f() {}
void h() {}
// void k() {i++;} // (3)
// void l() {j++;} // (4)

int m;
}
```

Select the one right answer:
(a) (1)
(b) (2)
(c) (3)
(d) (4)

Questions 12 Which of these statements are true?
Select all valid answers:
(a) A static method can call other non-static methods in the same class by using the this keyword.
(b) A class may contain both static and non-static variables and both static and non-static methods.
(c) Each object of a class has its own instance of each static member variable.
(d) Instance methods may access local variables of static methods.
(e) All methods in a class are implicitly passed a this parameter when called.

Questions 13 Which of these statements are true?
Select all valid answers:
(a) Transient variables will not be saved during serialization.
(b) Constructors can be declared abstract.
(c) The initial state of an array object constructed with the statement int a[] = new int[10] will depend on whether the variable a[] is a local variable, or a member variable of a class.
(d) Any subclass of a class with an abstract method must implement a method body for that method.
(e) Only static methods can access static members.

Questions 14 Which of the following are Java modifiers?
(a) public
(b) private
(c) friendly
(d) transient
(e) vagrant

Questions 15 Why might you define a method as native?
(a) To get to access hardware that Java does not know about.
(b) To define a new data type such as an unsigned integer.

(c) To write optimised code for performance in a language such as C/C++
(d) To overcome the limitation of the private scope of a method

Questions 16 What will happen when you attempt to compile and run this code?

```
public class Mod{
public static void main(String argv[]){
}
public static native void amethod();

}
```

(a) Error at compilation: native method cannot be static.
(b) Error at compilation native method must return value.
(c) Compilation but error at run time unless you have made code containing native amethod available.
(d) Compilation and execution without error.

Questions 17 What will happen when you attempt to compile and run this code?

```
private class Base{}
public class Vis{
transient int iVal;
public static void main(String elephant[]){
}
}
```

(a) Compile time error: Base cannot be private.
(b) Compile time error indicating that an integer cannot be transient.
(c) Compile time error transient not a data type.
(d) Compile time error malformed main method.

Questions 18 What will happen when you attempt to compile and run the following code?

```
public class Hope{
public static void main(String argv[]) {
Hope h = new Hope();
}

protected Hope() {
for(int i =0; i <10; i ++){
System.out.println(i);
}
}
}
```

(a) Compilation error: Constructors cannot be declared protected.
(b) Run time error: Constructors cannot be declared protected.
(c) Compilation and running with output 0 to 10.
(d) Compilation and running with output 0 to 9.

Questions 19 What will happen when you attempt to compile and run the following code with the command line "hello there"?

```
public class Arg{
String[] MyArg;
public static void main(String argv[]){
```

```
MyArg=argv;
}

public void amethod(){
System.out.println(argv[1]);
}
}
```

(a) Compile time error.
(b) Compilation and output of "hello".
(c) Compilation and output of "there".
(d) None of the above.

Questions 20 Which of the following method will not give you any error when placed on line no.3? Select one correct answer.

```
1 interface i2
2 {
3 //Here
4 }
```

(a) static void abc();
(b) abstract void abc();
(c) native void abc();
(d) synchronized void abc();
(e) final void abc();
(f) private void abc();
(g) protected void abc();

Answers:1 (c)
The [] notation can be placed both before and after the variable name in an array declaration. Multidimensional arrays are created by creating arrays that can contain references to other arrays. The statement new int[4][] will create an array of length 4, which can contain references to arrays of int values. The statement new int[4][4] will create the same array, but will also create four arrays, each containing four int values. References to each of these arrays are stored in the first array. The statement int [][4] will not work, because the dimensions must be created from left to right. Extra spaces are not significant.

Answers:2 (b), (e)
The size of the array cannot be specified as in (b), (e). The size of the array is given implicitly by the initialization code. The sizer of the array is never given during the declaration of an array reference. The size of an array is always associated with the array instance, not the array reference.

Answers:3 (a), (b), (e)
Non-static methods have an implicit this object reference. The this reference is not a normal reference variable that can be changed in the way attempted by statement (c). The this reference can be used to refer to both object and class members within the current context. However, it cannot be used to refer to local variables in the way attempted by statement (d).

Answers:4 (a), (d)
The first and third pairs of methods will compile correctly. The second pair of methods will not compile correctly, since their method signatures do not differ and the compiler has therefore no way of diffrentiating between the two methods. Note that return type and the names of the parameters are

not a part of the method signatures. Both methods in the first pair named fly and therefore overload this method name. The methods in pair three do not overload the method name glide, since only one method has the name. The method name Glide is distinct from the method name glide, as identifiers in Java are case-sensitive.

Answers:5 (a)
A constructor does not declare any return type, not even void. A constructor cannot be final, static or abstract.

Answers:6 (b), (e)
A constructor can be declared private, but this means that this constructor can only be used directly within the class. Constructors need not initialize all the member variables in a class. A member variable will be assigned a default value if not explicitly initialized. A constructor is non-static, and as such it can access directly both the static and non-static members of the class.

Answers:7 (c)
A compilation error will be encountered in (3), since the class does not have a constructor accepting a single argument of type int. The declaration at (1) declares a method, not a constructor, since it have a return type. The method happens to have the same name as the class, but that is irrelevant. The class has an implicit default constructor since the class contains no constructor declarations. This allows the instantiation at (2) to work.

Answers:8 (c), (d)
A class or interface name can be referred to by using either its fully qualified name or its simple name. Using the fully qualified name will always work, but in order to use the simple name it has to be imported. By importing net.basemaster.* all the type names from the package net.basemaster will be imported and can now be referred to using simple names. Importing net.* will not import the subpackage basemaster.

Answers:9 (C)
A class is uninstantiable if the class is declared abstract. The declararion of an abstract method cannot provide an implementation. The declaration of a non-abstract method must provide an implemenation. If any method in a class is declared abstract, then the whole class must be declared abstract. Definition (d) is not valid, since it omits the class keyword.

Answers:10 (b), (e)
You cannot specify visibility of local variables. They are accessible only within the block in which they are declared. Objects themselves do not have any visibility, only the references to the object. If no visibility modifier(public, protected or private) is given in the member declaration of a class, the member is only accessible to classes in the same package. A class does not have access to members of a superclass with default accessibility, unless both classes are in the same package. Inheritance has no consequence with respect to accessing members wit hdefaul;t accessibility. Local variables acannot be declared static and cannot be given an accessibility modifier.

Answers:11 (c)
The line "void k() { i++; }" can be reinserted without introducing errors. Reinserting line (1) will cause the compilation to fail, since MyOtherClass will try to override a final method. Reinserting line(2) will fail since MyOtherClass will no longer have a default constructor. The main() method needs a constructor that takes zero arguments. Reinserting line (3) will work without any problems, but reinserting line (4) will fail, since the method will try to access a private member of the superclass.

Answers:12 (b)
The keyword this can be only used in non-static methods. Only one instance of each static member variable of a class is created. This instance is shared among all objects of the class. Local variables are only accessible within the local scope, regardless of whether the local scope is defined within a static method.

Answers:13 (a)
The transient keyword signifies that the variables should not be stored when the object are serialized. Constructors cannot be declared abstract. Elements in an unitialized array object get the default value corresponding to the type of the elements. Whether the reference variable pointing to the array object is a local or a member variable does not matter. Abstract methods from a superclass need not be implemented by abstract subclass.

Answers:14 (a), (b), (d)
The keyword transient is easy to forget as is not frequently used, and it is considered in the access modifiers because it modifies the behaviour of the object.

Answers:15 (a), (c)

Answers:16 (d)
It would cause a run time error if you had a call to amethod though.

Answers:17 (a)
A top level (non nested) class cannot be private.

Answers:18 (d)

Answers:19 (a)
You will get an error saying something like "Cant make a static reference to a non static variable". Note that the main method is static. Even if main was not static the array argv is local to the main method and would thus not be visible within amethod.

Answers:20 (b)
Interface methods can't be native, static, synchronized, final, private, or protected.

Operator and assignments

Questions .1 Which statements about the output of the following program are true?

```
public class Logic {
public static void main(String args[]) {
int i = 0;
int j = 0;

boolean t = true;
boolean r;

r = (t && 0<(i+=1));
r = (t && 0<(i+=2));
r = (t && 0<(j+=1));
```

```
r = (t || 0<(j+=2));

System.out.println( i + " " + j );
}
}
```

(a) The first digit printed is 1.
(b) The first digit printed is 2.
(c) The first digit printed is 3.
(d) The second digit printed is 1.
(e) The second digit printed is 2.
(f) The second digit printed is 3.

Questions 2 Which statements about the output of the following program are true?

```
public static void main(String args[])
{
int i =0;
i = i++;
System.out.println(i);
}
```

(a) 0 is printed.
(b) 1 is printed.

Questions 3 Which statements about the output of the following program are true?

```
public class EqualTest {
public static void main(String args[]) {
String s1 = "YES";
String s2 = "YES";
if ( s1 == s2 ) System.out.println("equal");
String s3 = new String("YES");
String s4 = new String("YES");
if ( s3 == s4 ) System.out.println("s3 eq s4");
}
}
```

(a) "equal" is printed, "s3 eq s4" is printed.
(b) "equal" is printed only.
(c) "s3 eq s4" is printed only.
(d) Nothing is printed.

Questions 4 What happens when you try to compile and run the following code?

```
public class EqualsTest {
public static void main(String args[]) {
char A = '\u0005';
if ( A == 0x0005L )
System.out.println("Equal");
else
System.out.println("Not Equal");
}
}
```

(a) The compiler reports "Invalid character in input" in line 3

(b) The program compiles and prints "Not equal".
(c) The program compiles and prints "Equal".

Questions 5 What will happen when you attempt to compile and run the following code?

```
public class As{
int i = 10;
int j;
char z= 1;
boolean b;

public static void main(String argv[]){
As a = new As();
a.amethod();
}

public void amethod(){
System.out.println(j);
System.out.println(b);
}
}
```

(a) Compilation succeeds and at run time an output of 0 and false.
(b) Compilation succeeds and at run time an output of 0 and true.
(c) Compile time error b is not initialised.
(d) Compile time error z must be assigned a char value.

Questions 6 Given the following code what will be the output?

```
class ValHold {
public int i = 10;
}

public class ObParm {
public static void main(String argv[]) {
ObParm o = new ObParm();
o.amethod();
}

public void amethod(){
int i = 99;
ValHold v = new ValHold();
v.i=30;
another(v,i);
System.out.println(v.i);
} //End of amethod

public void another(ValHold v, int i){
i = 0;
v.i = 20;
ValHold vh = new ValHold();
v = vh;
System.out.println(v.i+ " "+i);
```

}//End of another
}
(a) 10,0,30
(b) 20,0,30
(c) 20,99,30
(d) 10,0,20

Questions 7 Here are three proposed alternatives to be used in a method to return false if the object reference x has the null value. Which statement will work?
(a) if (x == null) return false;
(b) if (x.equals(null)) return false;
(c) if (x instanceof null) return false;

Questions 8 What will be the result of compiling and running the given program?
Select one correct answer.

```
1 class Q1
2 {
3 public static void main(String arg[])
4 {
5 int a[]={2,2};
6 int b=1;
7 a[b]=b=0;
8 System.out.println(a[0]);
9 System.out.println(a[1]);
10 }
11 }
```

(a) Compile time error at the line no. 5.
(b) Run time error at the line no. 5.
(c) Program compiles correctly and print 2,0 when executed.
(d) Program compiles correctly and print 0,2 when executed.

Questions 9 What will be the result of compiling and running the given program?
Select one correct answer.

```
1 public class Q4
2 {
3 public static void main(String[] args)
4 {
5 boolean t1 = true, t2 = false, t3 = t1;
6 boolean t = false;
7 t &&= (t1 || ( t2 && t3));
8 System.out.println(t);
9 }
10 }
```

(a) Program compiles correctly and print true when executed.
(b) Program compiles correctly and print false when executed.
(c) Compile time error.
(d) Run time error.

Questions 10 What will be the result of compiling and running the given program?
Select one correct answer.

```
1 class AA{}
2 class BB extends AA{}
3 class Q6
4 {
5 public static void main(String arg[])
6 {
7 AA a=null;
8 BB b=(BB)a;
9 System.out.println(b);
10 System.out.println(b instanceof BB);
11 System.out.println(b instanceof AA);
12 }
13 }
```

(a) Program compiles correctly and print null,true,false.
(b) Program compiles correctly and print null,true,true.
(c) Program compiles correctly and print null,false,false.
(d) Program compiles correctly and print null,false,true.
(e) Compile time error at line no.8 as null value cannot be casted.
(f) Run time error at line no. 8 as null value cannot be casted.

Questions 11 Which one of the following are valid character contants?
Select any two.
(a) char c = '\u000d' ;
(b) char c = '\u000a';
(c) char c = '\u0000';
(d) char c = '\uface' ;

Answers: **1**. **(c), (d)**
Unlike & and | operators, the && and || operators short circuit the evaluation of their operands if the result of the operation can be determined just based on the value of the first operand. The second operand of the || operation in the program is never evaluated. Variable i ends up with the value 3 which is the first digit printed, and j ends up with a value of 1 which is the second digit printed.

Answers: 2 (a)
To explain u in a more simple way...

int k = 10;
int j;

Post increment :
j = k++;

In this statement the value of k is 10, & there is a ++ operator also attached with it, so it will be assigned to j, so now j gets 10, but k becomes as 11 because the ++ expression comes after the variable, or if u see k as k++.

Pre-increment :
j = ++k;
In this case, k is having 10 as value, then the pre-increment operation is being performed, so k becomes 11 in this case, & then it is being assigned to j, so now j becomes 11 & k also becomes 11.

The thing which u have must have noticed in the post increment & pre-increment is that the value of k is increased by 1, where as j in the first case becomes 10 & in second case j becomes 11.

Answers: 3 (b)
The compiler creates one String object for both s1 and s2, thus "equal" appears. But using the new String operator two distinct objects are created so "s3 eq s4" does not appear.

Answers: **4 (c)**
The compiler promotes variable A to a long before the comparison. The compiler does not report "Invalid character in input" in line 3 because this is correct form for initializing a char primitive. Therefore, answer (a) is incorrect. Because (b) cannot possibly be correct.⇒(c) is correct

Answers: **5 (a)**
The default value for a boolean declared at class level is false, and integer is 0, and 1 is within the range of the char and don't need casting.

Answers: 6 (d)
In the call another(v,i);
A COPY of the reference to v is passed and thus any changes will be intact after this call.

Answers: 7 (a)
The ONLY correct way to check a reference for the null value. Answer (b) is incorrect because if x is null, there will not be an object whose equals method can be called. This statement would cause a NullPointerException at runtime when x is null. Answer (c) is incorrect because only a reference type, such as a class, or array, can be the right operand of the instanceof operator.

Answers: 8 (c)
First of all value of b which is 1 is assigned to array index then rest of the code will be executed.

Answers: 9 (c)
Compile time error: There is no operator like &&=

Answers: 10 (c)
As null value can be casted but do remember it is of no use to cast null value that is why it is written in Khalid Mughal's book that we can not cast null value.Secondly,instanceof operator always return false for null value.

Answers: **11 (c), (d)**
'\u000d' and '\u000a' are not valid as these value do line break. Note:'\u000a' is not valid even when used behind single line comments (i.e. //.....),but offcourse for multiline comment it is valid. '\uface' is valid because we can use any character from range 0 - 9 or A - F or a - f.

Language fundamentals

Questions 1 Which of the following lines are valid declarations?
(a) char a = '\u0061';
(b) char \u0061 = 'a';
(c) ch\u0061r a = 'a';
(d) char a = (char) 65;

Questions 2 Is an empty file a valid source file?
(a) True.
(b) False.

Questions 3 Which of these are valid declaration of the main() method?
(a) static void main(String args[]) {/* … */}
(b) public static int main(String args[]) {/* … */}
(c) public static void main(String args) {/* … */}
(d) final static public void main(String[] arguments) {/* … */}
(e) public int main(String args[], int argc) {/* … */}
(f) public void main(String args[]) {/* … */}

Questions 4 Which one of the following are not valid character contants?
Select any two.
(a) char c = '\u00001' ;
(b) char c = '\101';
(c) char c = 65;
(d) char c = '\1001' ;

Answers: 1 (a), (b), (c), (d)
All are valid declarations. The \uxxxx notation can be used anywhere in the source to represent Unicode characters, and also casted integer to a char primitive type.

Answers: 2 (a)
Although nonsensical, an empty file ia a valid source file. A source file can contain an optional package declaration, any number of import statements and any number of class and interface definitions.

Answers: 3 (d)
A valid declaration of the main() method must be public and static, have void as return type and take a single array of String objects as arguments. The order of the static and public keywords is irrelevant. Also, declaring the method final does not affect the method's potential to be used as a main() method.

Answers: 4 (a), (d)
You cannot use five digits after \u."char c = 65" is valid because it will take it as ascii value. '\101' is representing a octal value which is equivalent to 65 which is valid but '\1001' is not valid as its decimal value is 513 and you can give only those values which represent 0 to 255 in decimal.

Object Oriented programming

Questions 1 Assume we have the following code in the file /abc/def/Q.java:

```
//File: /abc/def/Q.java:
package def;

public class Q {
private int privateVar;
int packageVar;
protected int protectedVar;
public int publicVar;
}
```

And this code is in /abc/Tester.java:

```
//File: /abc/Tester.java:
import def.Q;

public class Tester extends Q {
Q sup = new Q();
Sub sub = new Sub();

public void someMethod() {
// First, try to refer to sup's memebers.
sup.privateVar = 1; // Line 1
sup.packageVar = 2; // Line 2
sup.protectedVar = 3; // Line 3
sup.publicVar = 4; // Line 4

// Next, try to refer to this object's members
// supplied by class Q.
privateVar = 5; // Line 5
packageVar = 6; // Line 6
protectedVar = 7; // Line 7
publicVar = 8; // Line 8

// Next, let's try to access the members of
// another instance of Tester.
Tester t = new Tester();
t.privateVar = 9; // Line 9
t.packageVar = 10; // Line 10
t.protectedVar = 11; // Line 11
t.publicVar = 12; // Line 12

// Finally, try to refer to the members in a
// subclass of Tester.
sub.privateVar = 13; // Line 13
sub.packageVar = 14; // Line 14
sub.protectedVar = 15; // Line 15
sub.publicVar = 16; // Line 16
}
}
```

And this code is in /abc/Sub.java:
//File: /abc/Sub.java:
public class Sub extends Tester {
}
Assume the directory, /abc, is in the compiler's CLASSPATH.
When you try to compile Tester.java there will be several compiler errors. For each of the labeled lines above, decide whether or not a compiler error will be generated.

Questions 2 Given the following listing of the Widget class:
class Widget extends Thingee { //(1)
static private int widgetCount = 0; //(2)
public String wName; //(3)
int wNumber; //(4)
static synchronized int addWidget() { //(5)
wName = "I am Widget # " + widgetCount; //(6)
return widgetCount; //(7)
} //(8)
public Widget() { //(9)
wNumber = addWidget(); //(10)
} //(11)
} //(12)
What happens when you try to compile the class and use multiple Widget objects in a program?
(a) The class compiles and each Widget will get a unique wNumber and wName reflecting the order in which the Widgets were created.
(b) The compiler objects to line 6.
(c) The class compiles, but a runtime error related to the access of the variable wName occurs in the addWidget method.

Questions 3 The following method definition is designed to parse and return an integer from an input string that is expected to look like "nnn,ParamName." In the event of a NumberFormatException, the method is to return –1.
public int getNum (String s) { //(1)
try { //(2)
String tmp = S.substring(0, S.indexOf(',')); //(3)
return Integer.parseInt(tmp); //(4)
} catch (NumberFormatException e) { //(5)
System.out.println("Problem in " + tmp); //(6)
} //(7)
return –1; //(8)
} //(9)
What happens when you try to compile this code and execute the method with an input string that does contain a comma separating the number from the text data?
(a) A compiler error in line 6 prevents compilation.
(b) The method prints the error message to standard output and returns –1
(c) A NullPointerException is thrown in line 3.
(d) A StringIndexOutOfBoundsException is thrown in line 3.

Questions 4 Your chief Software designer has shown you a sketch of the new Computer parts system she is about to create. At the top of the hierarchy is a Class called Computer and under this are two child classes. One is called LinuxPC and one is called WindowsPC.

The main difference between the two is that one runs the Linux operating System and the other runs the Windows System (of course another difference is that one needs constant re-booting and the other runs reliably). Under the WindowsPC are two Sub classes one called Server and one Called Workstation. How might you appraise your designers work?
(a) Give the goahead for further design using the current scheme.
(b) Ask for a re-design of the hierarchy with changing the Operating System to a field rather than Class type.
(c) Ask for the option of WindowsPC to be removed as it will soon be obsolete.
(d) Change the hierarchy to remove the need for the superfluous Computer Class.

Questions 5 Given the following class definition which of the following can be legally placed after the comment line //Here ?

```
class Base{
public Base(int i){}
}
public class MyOver extends Base {
public static void main(String arg[]) {
MyOver m = new MyOver(10);
}
MyOver(int i){
super(i);
}
MyOver(String s, int i){
this(i);
//Here
}
}
```

(a) MyOver m = new MyOver();
(b) super();
(c) this("Hello",10);
(d) Base b = new Base(10);

Questions 6 Suppose you have two classes defines as follows:
class ApBase extends Object implements Runnable
class ApDerived extends ApBase implements Observer
Given two variables created as follows:
ApBase aBase = new ApBase();
ApDerived aDer = new ApDerived();
Which of the following Java statements will compile and execute without error?
(a) Runnable rn = aDer;
(b) Runnable rn2 = (Runnable) aBase;
(c) Observer ob = aBase;
(d) Observer ob2 = (Observer) aBase;

Questions 7 Suppose we have an ApBase class declared as:
class ApBase extends Object implements Runnable
The following code fragment takes a reference to an ApBase object and assigns it to a variety of variables. What will happen when you try to compile and run this code?
ApBase aBase = new ApBase();
Runnable aR = aBase;

Object obj = aR;
ApBase x = (ApBase)obj;
(a) The compiler objects to line 2
(b) The compiler objects to line 3
(c) The code compiles but when run throws a ClassCastException in line 4
(d) The code compiles and runs without problem.

Questions 8 Suppose you have two classes defines as follows:
class ApBase extends Object implements Runnable
class ApDerived extends ApBase implements Observer
Given two variables created as follows:
ApBase aBase = new ApBase();
ApDerived aDer = new ApDerived();
Which of the following Java statements will compile and execute without error?
(a) Object obj = aBase; Runnable rn = obj;
(b) Object obj = aBase; Runnable rn = (Runnable)obj;
(c) Object obj = aBase; Observer ob = (Observer)aBase;
(d) Object obj = aDer; Observer ob2 = obj;

Questions 9 What will happen if you attempt to compile and run the following code?

```
class Base {}
class Sub extends Base {}
class Sub2 extends Base {}

public class CEx {
public static void main(String argv[]){
Base b = new Base();
Sub s = (Sub) b;
}
}
```

(a) Compile and run without error.
(b) Compile time Exception.
(c) Runtime Exception.

Questions 10 You have these files in the same directory. What will happen when you attempt to compile and run Class1.java if you have not already compiled Base.java ?

```
//Base.java
package Base;
class Base {
protected void amethod() {
System.out.println("amethod");
}
}

//Class1.java
package Class1;
public class Class1 extends Base {
public static void main(String argv[]){
Base b = new Base();
b.amethod();
```

```
}
}
```
(a) Compile Error: Methods in Base not found
(b) Compile Error: Unable to access protected method in base class
(c) Compilation followed by the output "amethod"
(d) Compile error: Superclass Class1.Base of class Class1.Class1 not found

Questions 11 You are taking over an aquarium simulation project. Your predecessor had created a generic Fish class that includes an oxygenConsumption method declared as follows:
public float oxygenConsumption(float temperature)
The aquarium simulation sums oxygen consumption for all fish in the tank with the following code fragment, where fishes is an array of Fish object references:

```
float total = 0;
for (int i = 0; i < fishes.length; i++) {
total += fishes[i].oxygenConsumption(t);
}
```

you are writing a subclass for a particular fish species. Your task is to provide a method with species-specific metabolism data that will transparently fit into the simulation. Do you want to overload or override the oxygenConsumption method?
(a) overload.
(b) override.

Questions 12 The GenericFruit class declares the following method to return a float number of calories in the average serving size: [3]
public float aveCalories()
Your Apple class, which extends GenericFruit, overrides this method. In a DietSelection class that extends Object, you want to use the GenericFruit method on an Apple object. Select the correct way to finish the statement in the following code fragment so the GenericFruit version of aveCalories is called using the gf reference, or select option (d)

```
GenericFruit gf = new Apple();
float cal = // finish this statement using gf
```

(a) gf.aveCalories();
(b) ((GenericFruit)gf).aveCalories();
(c) gf.super.aveCalories();
(d) There is no way to call the GenericFruit method.

Questions 13 What will be the result of compiling and running the given program?
Select one correct answer.

```
1 public class Child extends Parent
2 {
3 public static int test(int i)
4 {
5 return 20;
6 }
7 public static void main(String[] args)
8 {
9 Parent c = new Child();
10 System.out.println(c.test(10));
11 }
12 }
```

```
13 class Parent
14 {
15 public static int test(int i)
16 {
17 return 5;
18 }
19 }
```
(a) Compile time as we can't overide static methods.
(b) Run time error as we can't overide static methods.
(c) Program compiles correctly and prints 5 when executed.
(d) Program compiles correctly and prints 20 when executed.

Questions 14 What will be the result of compiling and running the given program?
Select one correct answer.
```
1 class sample
2 {
3 sample(String s)
4 {
5 System.out.println("String");
6 }
7 sample(Object o)
8 {
9 System.out.println("Object");
10 }
11 }
12 class constructor
13 {
14 public static void main(String arg[])
15 {
16 sample s1=new sample(null);
17 }
18 }
```
(a) Compile time error as call to constructor at line no. 16 is ambigious.
(b) Run time error as call to constructor at line no. 16 is ambigious.
(c) Program compiles correctly and prints "object" when executed.
(d) Program compiles correctly and prints "string" when executed.

Questions 15 What will be the result of compiling and running the given program?
Select one correct answer.
```
1 class sample
2 {
3 sample(String s)
4 {
5 System.out.println("String");
6 }
7 sample(StringBuffer sb)
8 {
9 System.out.println("StringBuffer");
10 }
11 }
```

```
12 class constructor
13 {
14 public static void main(String arg[])
15 {
16 sample s1=new sample(null);
17 }
18 }
```

(a) Compile time error as call to constructor at line no. 16 is ambigious.
(b) Run time error as call to constructor at line no. 16 is ambigious.
(c) Program compiles correctly and prints "StringBuffer" when executed.
(d) Program compiles correctly and prints "string" when executed.

Questions 16 What will be the result of compiling and running the given program?
Select one correct answer.

```
1 class A
2 {
3 void callme()
4 {
5 System.out.println("A");
6 }
7 int r=10;
8 }
9 class B extends A
10 {
11 public void callme()
12 {
13 System.out.println("B");
14 }
15 int r=20;
16 }
17 class Q16
18 {
19 public static void main(String args[])
20 {
21 A a = new B();
22 a.callme();
23 System.out.println(a.r);
24 }
25 }
```

(a) Compile time error.
(b) Program compiles correctly and prints "A" and 10 when executed.
(c) Program compiles correctly and prints "B" and 20 when executed.
(d) Program compiles correctly and prints "B" and 10 when executed.

Questions 17 Whether the following code compile or not?
Select any two.

```
1 class Base {}
2 class Agg extends Base
3 {
4 public String getFields()
```

```
5 {
6 String name = "Agg";
7 return name;
8 }
9 }
10 public class Inheritence
11 {
12 public static void main(String argv[])
13 {
14 Base a = new Agg();
15 System.out.println(a.getFields());
16 }
17 }
```

(a) It will compile.
(b) It will not compile.
(c) It will compile if we cast the variable a for the object of class Agg like ((Agg) a).getFields()
(d) We can cast the variable a for the object of class Agg, but it is not required.

Answers: 1 When one class extends another class, instances of the subclass will contain their own copies of ALL members declared in the superclass (including any private variables). In the example above all instances of Tester will contain all four of the variables declared in Q. However, not all of these variables will be accessible (i.e., visible) by instances of Tester.
In the descriptions below, we say a variable is inherited by a subclass if it is accessible to instances of the subclass.
sup.privateVar = 1; // Line 1
private members are only accessible from within the class in which they are declared. Because privateVar was declared in class Q, it is not accessible from class Tester. The following compiler error results:
Tester.java:10: privateVar has private access in def.Q
sup.privateVar = 1; // Line 1
^
sup.packageVar = 2; // Line 2
package members are only accessible to classes within the same package as the class that declares them. In the case, the member packageVar, is declared in class Q, in the def package. package is accessible only from within classes in the def package. Since the Tester class is in the "default package" it does not have access to packageVar. The following compiler error results:
Tester.java:11: packageVar is not public in def.Q; cannot be accessed from outside package
sup.packageVar = 2; // Line 2
^
sup.protectedVar = 3; // Line 3
Even though class Tester extends class Q, instances of Tester are not allowed access to protected members in instances of Q, proper. Because sup is an instance of Q, proper, a Tester instance will not have access to its protected members. An instance of class Tester can only access protected members in instances of classes that are of type Tester. This includes classes that inherit from Tester (such as the Sub class). The following compiler error results:
Tester.java:12: protectedVar has protected access in def.Q
sup.protectedVar = 3; // Line 3
^
sup.publicVar = 4; // Line 4

public members are available "everywhere". Consequently, Tester's reference to sup.publicVar compiles without error.

privateVar = 5; // Line 5
private members are only accessible from within the class in which they are declared. Because privateVar was declared in class Q, it is not accessible from class Tester. Note that on Line 1 we were trying to refer to a private member in another object (the member sup). On line 5 we are trying to refer the copy of privateVar that Tester contains by virtue of being a subclass of Q. In terms of the visibility of privateVar, however, it does not matter. privateVar is a private member declared in the Q class and can ONLY be accessed from within that class. The following compiler error results:
Tester.java:17: privateVar has private access in def.Q
privateVar = 5; // Line 5
^
packageVar = 6; // Line 6
As was true for line 2, package members are only accessible to classes within the same package as the class that declares them. Here we are attempting to access the packageVar member contained in an instance of the Tester class. In other words, an instance of Tester is trying to access the packageVar it contains by virture of its inheritance from Q. The protectedVar member was declared by a class in the def package and Tester is not in that package. Therefore, Tester cannot access the packageVar member. The following compiler error results:
Tester.java:18: packageVar is not public in def.Q; cannot be accessed from outside package
packageVar = 6; // Line 6
^
protectedVar = 7; // Line 7
There is no compiler error for line 7.
An instance of Tester can refer to its own copy of protectedVar, the copy of protectedVar contained in some other instance of Tester, or the copy of protectedVar contained in an instance of a subclass of Tester. An instance of Tester cannot, however, refer to a copy of protectedVar in an object that is not an instance of Tester.

publicVar = 8; // Line 8
As was the case with line 12, public members are available "everywhere". Consequently, Tester's reference to publicVar compiles without error.

t.privateVar = 9; // Line 9
As was the case with lines 1 and 5, the member, privateVar, is declared in class Q and can therefore only be accessed by instances of class Q, proper (i.e., non-subclasses of class Q). The following compiler error results:
Tester.java:25: privateVar has private access in def.Q
t.privateVar = 9; // Line 9
^
t.packageVar = 10; // Line 10
As was the case with lines 2 and 6, the member, packageVar, is declared in class Q which is in the def package. Only instances of classes in the same package as class Q can access the package variable, packageVar. Remember that because the Tester class does not have a package statement it is placed in Java's "default package". The following compiler error results:
Tester.java:26: packageVar is not public in def.Q; cannot be accessed from outside package
t.packageVar = 10; // Line 10
^
t.protectedVar = 11; // Line 11

You may want to refer back to the description of line 7. If we have a class (call it B) that contains a protected member (call it p) and another class (call it S) extends B, all instances of S can access protected members in any instance of class S, including subsclasses of S.
Once again, in this coded example, class B is class Q, class S is class Tester, and p is the protected member protectedVar. Line 11 is an example of an instance of Tester attempting to reference a protected member in another instance of Tester (represented by the member, t). Since an instance of Tester is attempting to reference a protected member in another instance of Tester, the line compiles cleanly.

t.publicVar = 12; // Line 12
As was the case with lines 4 and 8, public members are available "everywhere". Consequently, Tester's reference to t.publicVar compiles without error.

sub.privateVar = 13; // Line 13
As was the case with lines 1, 5, and 9, the member, sub.privateVar, is declared in class Q and can therefore only be accessed by instances of class Q, proper (i.e., non-subclasses of class Q). The following compiler error results:
Tester.java:32: privateVar has private access in def.Q
sub.privateVar = 13; // Line 13

sub.packageVar = 14; // Line 14
As was the case with lines 2, 6, and 10, the member, sub.packageVar, is declared in class Q which is in the def package. To access this member the class attempting the access (Tester, in this case) must be in the def package. Remember that since Tester does not have a package statement it is placed in Java's "default package". The following compiler error results:
Tester.java:33: packageVar is not public in def.Q; cannot be accessed from outside package
sub.packageVar = 14; // Line 14
^
sub.protectedVar = 15; // Line 15
You may want to refer back to the descriptions of lines 7 and 11. Actually, this line compiles cleanly for exactly the same reason line 11 compiled cleanly. Remember that class Sub extends class Tester. Consequently, any instances of Sub are also considered instances of Tester.

sub.publicVar = 16; // Line 16
As was the case with lines 4, 8 and 12, public members are available "everywhere". Consequently, Tester's reference to sub.publicVar compiles without error.

Answers: 2 (b)
The static method addWidget cannot access the member variable wName. Static methods can refer to static variables only, such as widgetCount.
Answers: 3 (a)
Because the scope of the tmp String is confined to the try block, thus, it cannot be used in line 6. answer (c) would not occur even if the scope of the tmp variable were fixed. answer (d) would occur only if the scope of the tmp variable were fixed by declaring and initializing tmp in the first line of the method.

Answers: 4 (b)
This question is about the requirement to understand the difference between the "is-a" and the "has-a" relationship. Where a class is inherited you have to ask if it represents the "is-a" relationship. As the difference between the root and the two children are the operating system you need to ask are Linux

and Windows types of computers.The answer is no, they are both types of Operating Systems. So option two represents the best of the options. You might consider having operating system as an interface instead but that is another story.
Of course there are as many ways to design an object hierarchy as ways to pronounce Bjarne Strousjoup, but this is the sort of answer that Sun will probably be looking for in the exam. Questions have been asked in discussion forums if this type of question really comes up in the exam. I think this is because some other mock exams do not contain any questions like this. I assure you that this type of question can come up in the exam. These types of question are testing your understanding of the difference between the is-a and has-a relationship in class design.

Answers: 5 (d)
Any call to this or super must be the first line in a constructor. As the method already has a call to this, no more can be inserted.

Answers: 6 (a), (b)
Answer (a) is correct because the ApDerived class inherits from ApBase, which implements Runnable. Answer (b) is also correct because the inserted cast (Runnable) is not needed but does not cause a problem. Answer (c) fails to compile because the compiler can tell that the ApBase class does not implement Observer. Answer (d) compiles, but fails to execute. Because of the specific cast the compiler thinks you know what you are doing, but the type of the aBase reference is checked when the statement executes and a ClassCastException is thrown.

Answers: 7 (d)
These casts and assignment are all legal. (a) is incorrect incorrect because an object reference can be assigned to an interface reference as long as the compiler knows that the object implements the interface. Answer (b) is incorrect because an interface reference can be assigned to a reference to Object because Object is the base of the Java class hierarchy. Answer (c) is incorrect because the object referred to has not lost its identity, so it passes the runtime cast check.

Answers: 8 (b)
It compiles and runs, the compiler assumes you know what you are doing with the cast to Runnable. Answer (a) fails to compile. As far as the compiler is concerned, obj is a plain Object so it objects to the assignment to a Runnable reference. Answer (c) compiles but fails to run. Because of the specific cast, the compiler thinks you know what you are doing, but the type of the aBase reference is checked when the statement executes, and a ClassCastException is thrown. Answer (d) fails to compile. As far as the compiler is concerned, obj is a plain Object so it objects to the assignment to an Observer reference.

Answers: 9 (c)
Without the cast to sub you would get a compile time error. The cast tells the compiler that you really mean to do this and the actual type of b does not get resolved until runtime. Casting down the object hierarchy as the compiler cannot be sure what has been implemented in descendent classes. Casting up is not a problem because sub classes will have the features of the base classes. This can feel counter intuitive if you are aware that with primitives casting is allowed for widening operations (ie byte to int).

Answers: 10 (d)
Using the package statement has an effect similar to placing a source file into a different directory. Because the files are in different packages they cannot see each other. The stuff about File1 not having

been compiled was just to mislead, java has the equivalent of an "automake", whereby if it was not for the package statements the other file would have been automatically compiled.

Answers: 11 (b)
by overriding the oxygenConsumption method, the Java runtime will call the overriding method for all fish where a specific method is provided or the generic method if there is none. Answer (a) is incorrect because if you overloaded the oxygenConsumption method using a different method signature, the Java runtime would not call the specific method for Fish where a specific method was provided. It would always call the generic method.

Answers: 12 (d)
There is no way for a class outside the GenericFruit hierarchy to call the GenericFruit method using an Apple reference. Answer (a) is incorrect because the runtime resolution of method calls finds the Apple method. Answer (b) is incorrect because this extra cast does not change the object type. Answer (c) does not create a valid Java statement.

Answers: 13 (c)
As both the method in class Child and Parent are static the method call is bound at compile time with the class whose reference variable is taken i.e. Parent.

Answers: 14 (d)
Whenever a method/Constructor has an argument which matches two different methods/Constructors defination then it will always call the most specific one. As in our case Object is a general class and is super class of all other classes so it will call the String version of the Constructor.

Answers: 15 (a)
As in this case both classes are peer i.e. both the classes is at the same level in the hierarchy chart.

Answers: 16 (d)
Method are bound at run time where as variables are bound at compile time.It is the type of object which will tell that which method is to be called(i.e. B in this case), but it is the type of reference which will tell you which variable is to be called(i.e. A in this case).

Answers: 17 (b), (c)
The program will not compile as the there is no method named getFields() defined in the class Base.It will compile if we cast the reference variable a for the object of class Agg like ((Agg) a).getFields()