**1)**

**Output**:

String

**Explanation** : In case of [method overloading](https://www.geeksforgeeks.org/overloading-in-java/), the most specific method is chosen at compile time. As ‘java.lang.String’ is a more specific type than ‘java.lang.Object’. In this case the method which takes ‘String’ as a parameter is choosen.

2) **Output:**

Compile Error at line 19.

**Explanation:** In this case of [method Overloading](https://www.geeksforgeeks.org/overloading-in-java/), the most specific method is choosen at compile time.  
As ‘java.lang.String’ and ‘java.lang.Integer’ is a more specific type than ‘java.lang.Object’,but between ‘java.lang.String’ and ‘java.lang.Integer’ none is more specific.  
In this case the Java is unable to decide which method to call.

**3)Output**:

abcd abc false

abcd abcd true

**Explanation :**In Java, String is immutable and string buffer is mutable.  
So string s2 and s1 both pointing to the same string abc. And, after making the changes the string s1 points to abcd and s2 points to abc, hence false. While in string buffer, both sb1 and sb2 both point to the same object. Since string buffer are mutable, making changes in one string also make changes to the other string. So both string still pointing to the same object after making the changes to the object (here sb2).

**4) Output:**

a

b

c

**Explanation:**  
While creating a new object of ‘Third’ type, before calling the default constructor of Third class, the default constructor of super class is called i.e, Second class and then again before the default constructor of super class, default constructor of First class is called. And hence gives such output.

**5) Output:**

10

200

400

**Explanation:**  
Since in ‘Second’ class it doesn’t have its own ‘i’, the variable is inherited from the super class. Also, the constructor of parent is called when we create an object of Second.

**6) Output**:

false, true, false

**Explanation:**  
The iterator() method returns an iterator over the elements in the list in proper sequence, it doesn’t return a List or a ListIterator object. A ListIterator can be obtained by invoking the listIterator method.

**7)Output:**

Run Time Exception

**Explanation:**  
Exception in thread “main” java.lang.IllegalThreadStateException at java.lang.Thread.start  
Thread cannot be started twice.

8) **Answer : 20**  
**Explanation :** Here the class instance variable name(num) is same as calc() method local variable name(num). So for referencing class instance variable from calc() method, [this](http://quiz.geeksforgeeks.org/this-reference-in-java/) keyword is used. So in statement **this.num = num \* 10**, num represents local variable of the method whose value is 2 and this.num represents class instance variable whose initial value is 100. Now in printNum() method, as it has no local variable whose name is same as class instance variable, so we can directly use num to reference instance variable, although this.num can be used.

**9)** **Answer :** C) The output is true and MyStuff does NOT fulfill the Object.equals() contract.  
**Explanation :** As equals(Object obj) method in Object class, compares two objects on the basis of equivalence relation. But here we are just confirming that the object is null or not, So it doesn’t fulfill [Object.equals()](https://www.geeksforgeeks.org/overriding-equals-method-in-java/) contract. As m1 is not null, true will be printed.

**10)** **Answer :** A) alpha beta b b  
**Explanation :** The statement **new Beta().go()**executes in two phases. In first phase Beta class constructor is called. There is no instance member present in Beta class. So now Beta class constructor is executed. As Beta class extends Alpha class, so call goes to Alpha class constructor as first statement by default(Put by the compiler) is **super()** in the Beta class constructor. Now as one instance variable(type) is present in Alpha class, so it will get memory and now Alpha class constructor is executed, then call return to Beta class constructor next statement. So alpha beta is printed.  
In second phase go() method is called on this object. As there is only one variable(type) in the object whose value is a. So it will be changed to b and printed two times. The [super keyword](http://quiz.geeksforgeeks.org/super-keyword/) here is of no use.

**11) Answer :** C) 4  
**Explanation :** *append(String str)* method,concatenate the str to *s1*. The *substring(int index)* method return the String from the given index to the end. But as there is no any String variable to store the returned string,so it will be destroyed.Now *indexOf(String s2)* method return the index of first occurrence of *s2*. So 4 is printed as s1=”JavaLove”.

**12) Answer :** B) Writing book  
**Explanation :** Since static methods can’t be overridden, it doesn’t matter which class object is created. As *a* is a*Author* referenced type, so always *Author* class method is called. If we remove *write()* method from *Author*class then *Writer*class method is called, as *Author*class extends *Writer* class.

**13)Output:**

Not equal

**Explanation:** Since, s1 and s2 are two different objects the references are not the same, and the == operator compares object reference. So it prints “Not equal”, to compare the actual characters in the string .equals() method must be used.

**14)Output:**

Inside static method, Person(whoAmI)

Kid(who)

Kid(whoAreYou)

**Explanation:** Static binding (or compile time) happens for static methods. Here *p.whoAmI()* calls the static method so it is called during compile time hence results in static binding and prints the method in *People* class.  
Whereas *p.whoAreYou()* calls the method in *Kid* class since by default Java takes it as a virual method i.e, dynamic binding.

**15)Output:**

gfg1gfg1gfg3

**Explanation :**Initially new Thread is started with name *gfg1* then in class Two the first run method runs the thread with the name *gfg1*, then after that a new thread is created calling run method but since a new thread can be created by calling start method only so the previous thread does the action and again *gfg1* is printed.Now a new thread is created by calling the start method so a new thread starts with *gfg3* name and hence prints *gfg3*.

**17) Output:**

Geeksforgeeks1

Geeksforgeeks2

**Explanation:**  
We know that static variables are called when a class loads and static variables are called only once. Now line 13 results to creation of object which inturn calls the constructor and “Geeksforgeeks” is printed second time.  
If in line 8 static variable would not have been used the object would have been called recursively infinitely leading to StackOverFlow error. See [this](https://ide.geeksforgeeks.org/wtntd4) for a sample run.

**18) Output:**

Static Block 1

Static Block 2

Value of num = 100

Value of mystr = Constructor

**Explanation:**  
Static block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which are executed in the same sequence in which they have been written into the program.  
**Note**: Static Methods can access class variables without using object of the class. Since constructor is called when a new instance is created so firstly the static blocks are called and after that the constructor is called. If we would have run the same program without using object, the constructor would not have been called.

**19) Output:**

Compilation fails.

Explanation:

The method calc() in class superClass is final and so cannot be overridden.

**20) Output:**

false

true

**Explanation:** In the source code of Integer object we will find a method ‘valueOf’ in which we can see that the range of the Integer object lies from IntegerCache.low(-128) to IntegerCache.high(127). Therefore the numbers above 127 will not give the expected output. The range of IntegerCache can be observed from the source code of the IntegerCache class.

**21) Output:**

* end of main
* **Explanation :** We know that [finalize()](https://www.geeksforgeeks.org/g-fact-24-finalfinally-and-finalize-in-java/) method is called by Garbage Collector on an object before destroying it. But here, the trick is that the str is String class object, not the Test class. Therefore, finalize() method of String class(if overridden in String class) is called on str. If a class doesn’t override finalize method, then by default Object class finalize() method is called.

**22) Output:**

finalize method called

end main

**Explanation :**  
When Garbage Collector calls finalize() method on an object, it **ignores** all the exceptions raised in the method and program will terminate normally.

**23) Answer :**

* 2
* **Explanation :**  
  Since t1 and t2 are local objects of m1() method, so they become eligible for garbage collection after complete execution of method unless any of them is returned.

**24)** Answer (c)

Explanation: When overriding a method of superclass, the method declaration in subclass cannot be more restrictive than that declared in the superclass.

**25)** Answer (d)  
Explanation : [(int… values) is passed as parameter to a method when you are not aware of the number of input parameter but know that the type of parameter](https://www.geeksforgeeks.org/variable-arguments-varargs-in-java/)(in this case it is int). When a new object is made in the main method, variable data is initialized to 5. A call to getData() method with attributes (7, 8 ,12), makes a call to the third getData() method, which increments the value of data variable by 2 and return 7.

**26)** Answer (d)  
**Explanation:** Since the method multiplier is marked as private, it isn’t inherited and therefore is not visible to the Derived.

**27)** Answer (d)  
**Explanation:** The catch statements are written in the order: more specific to more general. In the code above, a new exception of type Exception is thrown. First the code jumps to first catch block to look for exception handler. But since the IOException is not  
of the same type it is moves down to second catch block and finally to the third, where  
the exception is caught and 4 is printed. Therefore, the answer is 145, as the order  
of execution in terms of blocks is: try->catch->finally.

**28)** Answer (d)  
**Explanation:**static blocks in Java are executed even before the call to main is made by the compiler. In the main method, a new object of javaclass is made and its function() method is called which return 4. Number 5 is returned by a call to static function  
myMethod(). 4+5 = 9. Therefore, the output of the program is 1239, because 123 is printed  
on the console even before main method executes and main method on execution returns 9.

Note:- **The private method is getting called because the main method is actually inside that class whose object is created**

**29)** Ans: (a)

**Explanation:** A non-static variable can not be accessed in [static nested inner class](https://www.geeksforgeeks.org/inner-class-java/). “Nested” cannot access non-static variables[variable s in this case]. Therefore the error:

10: error: non-static variable s cannot be referenced from a static context

System.out.println(temp + s + value);

^

**30)** Ans: (b)

**31)** Ans: (d)

**32)** Ans: (b)

**Explanation:** == operator checks if two variable refer to the same object. Here a and b  
refers to two different objects. ?: is another form of if else statement that could be read as, condition : if true then do this : else do this.

**33)** Ans: (c)

**Explanation:** [Constructors](https://www.geeksforgeeks.org/constructors-in-java/) cannot be enclosed in try/catch block.

**34) Ans. (c)**  
**Explanation:** Priority queue always outputs the minimum element from the queue when remove() method is called, no matter what the sequence of input is.

**Here priority is similar to ranking, the lower the value, the better it is.**

**35) Ans. (c)**  
**Explanation:** A TreeSet sorts the data in ascending order that is inserted in it. Therefore, the output string contains all the strings arranged in ascending order. A TreeSet does not contain any duplicate element as it is a set. So in the output, there is just a single occurance of string ‘Geeks’.

**36) Ans. (d)**  
**Explanation:** list1.removeAll(list2) function deletes all the occurance of string in list2 from list1. Here, the string ‘Geeks’ appears in list2, so all the nodes of linked list in list1 that contains ‘Geeks’ as its data is removed from list1

**37)** **Ans. (d)**  
**Explanation:** An iterator made for iterating over Integer cannot be used to iterate over String data type. Corrected program : <https://ide.geeksforgeeks.org/DgeN0P>

**38)** **Ans.**(a)  
**Explanation:**A [private constructor](https://www.geeksforgeeks.org/private-constructors-and-singleton-classes-in-java/) cannot be used to initialize an object outside the class that it is defined within because it is no longer visible to the external class.

**39)** **Ans.**(c)  
**Explanation:**A constructor cannot be enclosed inside a try/catch block.

**40)** **Ans.**(a)  
**Explanation:**When a constructor is marked as private, the only way to create a new object of that class from some external class is using a method that creates a new object, as defined above in the program. The method create() is responsible for creation of Temp object from some other external class. Once the object is created, its method can be invoked from the class in which the object is created.

**41)** **Ans.**(a)  
**Explanation:**[Constructors can be chained](https://www.geeksforgeeks.org/constructor-chaining-java-examples/) and overloaded. When Test() is called, it creates another Test object calling the constructor Test(int temp).

**42)** **Ans.**(c)  
**Explanation:**Constructor call to super class must be the first statement in the constructor of the Derived class.

**43)** **Ans.**(d)  
**Explanation:**static inner classes cannot access non-static fields of the outer class.

**44)** **Ans.**(c)  
**Explanation:**LocalClass() method defines a local inner class. This method creates an object of class Inner and return the value of the variable data that resides within it.

**45)** **Ans.**(b)  
**Explanation:**the method getData() is undefined in Anonymous class which causes the compilation error.

**46)** **Ans.**(b)  
**Explanation:** Inner class defined above though, have access to the private variable data of the Outer class, but declaring a variable data inside an inner class makes it specific to the Inner class with no conflicts in term of variable declaration. For more see [Shadowing](https://docs.oracle.com/javase/tutorial/java/javaOO/localclasses.html#shadowing-and-local-classes)

**47)** **Ans.**(a)  
**Explanation:**[Nested Interfaces](https://www.geeksforgeeks.org/interface-nested-class-another-interface/) are defined in java. As both the interfaces has declaration of InnerMethod(), implementing it once works for both the InnerInterface and OuterInterface.

**48)** **Ans.** (c)  
**Explanation:**From the statement “thread.start()”, we have two threads [Main thread](https://www.geeksforgeeks.org/main-thread-java/) and “thread” thread. So either “GFG” can be printed or “Geeks”, depend on which thread, thread scheduler schedule.  
For (a), the parent thread after calling start() method is paused and the thread scheduler schedules the child thread which then completes its execution. Following this, the parent thread is scheduled. For (b), the parent thread calls start() method but continues its execution and prints on the console. When join() method is called, the parent thread has to wait for its child to complete its execution. Thread scheduler schedules child thread while the parent waits for the child to complete.

**49)** **Ans.**(d)  
**Explanation:**Invoking start() method on a thread moves the thread to a RUNNABLE state. But invoking start() method on a thread that has already started throws a IllegalThreadStateException because the thread is already in RUNNABLE state.