* When you try to access the character of the string which was not present in the string using charAt function, then StringIndexOutOfBoundsException is printed.
* The equals method compares for the value of equality.
* The == compares the reference not the value.
* Int[2][2] array2D;-🡪 Wrong way to define an array
* Int[][] array2D;🡪 recommended way to declare an array
* Int[]array2D[]🡪 coorect way but not legal to use.
* One of the bad practices are;
  + Checking for the array index count of bounds exception when iterating through an array to determine when all elements have been visited.
  + Checking for error and if necessary restarting the program to ensure that users are unaware problems.
  + Checking for array index out of bounds exception and ensuring that program can recover if one occur.
* When a class is extending another class then the methods which are ovveriding the super class methods should be of the same return type.
* An abstract class can contain both abstract and concrete methods
* An abstract class can also contain private members and private methods.
* An abstract class doesnot have to have a constructor without arguments.
* An abstract class doesnot have to have atleast one abstract method.
* But defining the abstract methods is illegal in the concrete class.
* The while and if only accepts the Boolean values that might be either true or false.
* The questions might be given to confuse you stating the normal values or incrementing I or j in the while of if loop which is wrong.
* The types that are valid in the switch case are int,Integer and the string type.
* And when you try to access the element which is not present in the array it will be throwing the exception ArrayIndexOutOfBoundsException.
* Never use the variable without declaring it. Like in the example;
* Int x=x2; int x2=j;
* Here in the above example, x2 is assigned to integer type of x and it was initialized later on in the next step.
* Pass leader question num 71,72,79,115,125,132,134 should be done.
* To add the content to the existing defined StringBuilder class the keyword append is used.
* The sb.delete(int startindex,int endindex) is used to delete the contents of the string builder. The start index includes the value but the last index excludes the last index value.
* SecurityException is one of the exception class in java.
* In the native For loop, when the condition that is the middle part evaluates to false, then the loop terminates as it is only evaluated only after each iteration through the loop.
* String replace is the valid function in java;

public String replace(char oldChar, char newChar)

public String replace(CharSequence target, CharSequence replacement)

* NullPointerException is a RuntimeException . In Java, a special **null** value can be assigned to an object reference. NullPointerException is thrown when an application attempts to use an object reference that has the null value. These include: Calling an instance method on the object referred by a null reference.
* In float, the declaration float f=100; is valid.
* We need to convert float type to double explicitly by casting not by implicit way of conversion.
* A particular user defined exception could be extended from any of the sub exception classes irrespective of extending it from the super exception class.
* The append function of the String Builder class only takes the one argument and it just appends the existing class with the particular object value.
* Whereas the insert of the String Builder class, can also insert the object value into the particular position of the defined String Builder class. The insert method takes the two arguments that is the index position and the value.
* In java, any class which is defined is implicitly defined by the Object class.
* By default Boolean value will be false.
* Example:-

Class App{

Private Boolean flag;

Public void dispResult(){

Int result=flag?5:10;

Sysout(“result is”+result);

}

Main(){

New App().dispResult();

}}

* In the above snippet, as flag is defined as the Boolean variable it can be used in the if condition as it return just true or false. And the above code returns false as by default Boolean value would be false and the answer is 10
* When a class extending another class, and the object of the child class is create with the parent class reference, then when some methods are overridden in the parent class are only called not the parent ones.
* Even the while condition return true or false not any other type.
* when a class is extending another class, the constructor in the super class should be only public
* Check out the 135 question from the pass leader. “Sexy question”.
* In java a method name can have same name as the fields.
* A class can have overloaded static methods in java.
* the fields need not to be initialized before.
* In the string methods like equals and the ==, if we directly compare the references without any data in it then the answer would be false in both the cases.
* Else it will return true if we compare the references with the data residing in it.
* A switch shouldn’t consist of any duplicate cases.
* Re-refer 143,211,218,224,241,242 again.(From pass leader)
* Abstract classes cant be instantiated but they can be subclassed.
* Semi colon after condition in while loop goes to check for the condition for infinite times if it is true, if false it will exit the loop.
* File not found exception is the sub class of IO.
* The catch clause is always throwable.
* Declaring a variable abstract is illegal in java
* If the variable is declared static with the reference type of the class the, by default that variable value would be null.
* Even when we are manually throwing the exception using throw and new key word, there again an object is created.
* FileNotFoundException is the checked exception.
* In java the methods can have same name as fields.
* A class can have overloaded methods.
* The fields can’t be used before initializing them.
* If the character type is not initialized when declaring the variable as instance then when we try to print the value of that character type, it doesn’t take any value(not even null).
* The default constructor of the sub class always invokes the no argument constructor of its super class.
* The continue statement skips the current iteration of a for, while, or do-while loop.
* An unlabeled break statement terminates the innermost switch, for, while, or do- while statement, but a labeled break terminates an outer statement.
* A class can only have one private constructor.
* sb.delete(0, sb.length());will empty the contents of a StringBuilder variable named sb.
* The break,continue cant be used outside the switch or for loops, not even in the if conditions.
* Benefits of using ArrayList over array in software development;
  + reduces memory footprint
  + dynamically resizes based on the number of elements in the list
* StringBuilder(int capacity)-- Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.
* Identifiers can begin with currency character, underscore or with a letter.
* After first character identifier can also contain digits and it could be of any length.
* In java, mian() method can be overloaed.
* A source code file can have only one public class and that should be matched with the filename.
* A class with no public modifier doesn’t have any rstrictions and naming conventions
* A class cannot be both abstract and final.
* A final class cannot be subclassed.
* An abstract class cannot be instantiated.
* The first concrete class to extend an abstract class must implement all of its abstract methods.
* An interface is by default abstract.
* An interface can have only abstract methods not the concrete methods.
* Interface methods are by default public and abstract.
* Final is the only access modifier available to the local variables.
* Local variables must be initiliazed before use.
* Final methods cannot be ovverriden in the sub class.
* Abstract methods ends with a semi colon as they do not contains body.
* The 1st non abstract class which is extending the super abstract class must implement all the abstract methods of the super abstract class.
* Abstract methods cannot be final or private
* A var-arg method can have only one argument method.
* In methods with normal parameters and a var-arg, the var-arg must come last.
* Declaring instance and the local with the same is called shadowing and this is legal in java.
* The transient and the volatile modifiers apply only to the instance variables.
* #--> not allowed.
* final methods cannot be overridden.
* Methods with an object reference return type can return a subtype.
* Methods with an interface return type can return any implementer.
* Abstract classes have constructors that are called when a concrete subclass is instantiated. Calls to this() and super() cannot be in the same constructor. You can have one or the other, but never both.
* If a class is extending its super class with invoking of the parameterized constructor using the object without the default constructor in the super class then you need to explicitly invoke the default constructor, otherwise the parameterized constructor in the sub class would be showing error.

class Bird {

{ System.out.print("b1 "); }

public Bird() { System.out.print("b2 "); }

}

class Raptor extends Bird {

static { System.out.print("r1 "); }

public Raptor() { System.out.print("r2 "); }

{ System.out.print("r3 "); }

static { System.out.print("r4 "); }

}

class Hawk extends Raptor {

public static void main(String[] args) {

System.out.print("pre ");

new Hawk();

System.out.println("hawk ");

}

* }

In the above code, the static block is called at the time of class loading and then when the hawk object is created the super classes are called and then the statements in the sop are printed first followed by the constructor block statements.

* Using the child class reference you can call the parent class methods as it is extending the parent but using the parent class reference you cannot call the child class methods. Explicit type casting should be there.
* Relational operators only result in true or false.
* The & and | operators always evaluate both operands.
* The ^ operator (called the "logical XOR") returns true if exactly one operand is true. instanceof is for reference variables only; it checks whether the object is of a particular type.
* The instanceof operator can be used only to test objects (or null) against class types that are in the same class hierarchy.
* For interfaces, an object passes the instanceof test if any of its superclasses implement the interface on the right side of the instanceof operator.
* In simpler terms, when a classs is extending another class and we are giving the instance of operator to the sub class reference which is extending the super class. This is valid only when that super class is implemented by the interface.
* Example is shown below,
* interface Vessel { }
* 4. interface Toy { }
* 5. class Boat implements Vessel { }
* 6. class Speedboat extends Boat implements Toy { }
* 7. public class Tree {
* 8. public static void main(String[] args) {
* 9. String s = "0";
* 10. Boat b = new Boat();
* 11. Boat b2 = new Speedboat();
* 12. Speedboat s2 = new Speedboat();
* 13. if((b instanceof Vessel) && (b2 instanceof Toy)) s += "1";
* 14. if((s2 instanceof Vessel) && (s2 instanceof Toy)) s += "2";
* 15. System.out.println(s);
* 16. }
* 17. }
* String methods use zero-based indexes, except for the second argument of substring().
* Strings have a *method* called length()—arrays have an *attribute* named length. switch statements can evaluate only to enums or the byte, short, int, char, and, as of Java 7, String data types. You can't say this:

Long s = 30;

switch(s) { }

* The do loop will enter the body of the loop at least once, even if the test condition is not met.
* Integer.parseInt(“two”) can throw a NumberFormatException, and IllegalArgumentException is its superclass (that is, a broader exception).