1. Why java is popular

* Robust
* Platform independent
* Lightweight
* No pointers => addresses
* OOPs concepts / principles
* Multithreaded
* Distributed environment
* Scalable
* Secured

1. Compilers and interpreters?

Compiler : check the complete code for syntax errors and if none then converts to some intermediate format

Hello.java => compiler => Hello.class (bytecode)

Interpreter -> read the code(.class) line by line and execute them to machine executable

1. JDK -> java dev kit [ comes up with the java libraries and all the excutable ]

JRE -> java runtime env [ provides with environment for java applications to execute and manage all java applications running]

JVM -> java virtual machine [ 1 instance of jvm per application ]

1. Packages : group of similar kind of classes

* Package names should all be small. And it is the 1st statement in the file except comments
* Class names every word should always start with uppercase

1. Diff between print and println
2. Data types - 8
   1. Byte
   2. Char
   3. Short
   4. Int
   5. Long
   6. Float
   7. Double
   8. Boolean – true or false
   9. String – derived data type
3. Variables
   1. Naming conventions
   2. Declaration and initialization
   3. For char wrap the values around ‘’
   4. For String wrap the values around “”
4. Operators
   1. Unary ++ -- - !
   2. Binary
      1. Arithmetic + - \* / %
      2. Relational < > <= >= == !=
      3. Logical & && | || So && and || are called short circuit operators
   3. Ternary ?:
5. Escape sequence : \n \t \” \’ \\
6. Conditional constructs
   1. If-else
   2. Switch-case
7. Looping constructs
   1. For
   2. While
   3. Do-while
   4. Break
   5. Continue
   6. Nested loops
8. Arrays
   1. collection of data of same type
   2. it is stored in continuous memory locations
   3. size once declared is fixed
   4. uses [] syntax to create arrays
   5. arrays support indexing that starts from index 0 to length-1
   6. Arrays have a length property that displays the total size of the array
   7. Depending on what type of array it is, it has a default value
      1. Primitives : primitive defaults ex: int 0, double 0.0
      2. Reference types like String or user defined => null
9. For Each loop
10. String
    1. == and equals
    2. In built methods
11. Class
    1. User defined data type
    2. Blueprint
    3. Composite data type
    4. Class can have data members and member method
12. Methods
    1. Methods can take input in the form of parameters in the ()
    2. Methods that do not return anything must be declared as void
    3. Methods give output by “return” keyword
    4. There can be more than 1 conditional return statements and only 1 of them is executed at any given point of time
13. Constructors
    1. is a special method with the name same as the class name
    2. they DO NOT have return types not even void
    3. They are automatically invoked when the object of the class is created using new
    4. They cannot be invoked using .
    5. They are invoked only the 1st time the object is created
    6. purpose is to initialize the data members of the class
14. Getters and Setters
    1. Getters are accessors with return type and no parameters  
       public int getEid(){return eid;}  
       For bolean properties method starts with is and not get
    2. Setters are modifiers with void return type and takes 1 parameter  
       public void setEid(int eid){this.eid = eid;}
15. toString()
    1. provides with the string representation of the object
16. Constructor Overloading : static polymorphism   
     More than 1 method with the same name, but with
    1. Different number of arguments
    2. Different types
    3. Different sequence
    4. Changing the return type or access specifier or method name is NOT OVERLOADING
    5. Can be done in same class or parent-child as well
17. Has-a
    1. Reference of a class is added in another class
    2. A{B b}; B{} => class A has a dependency on an object of class B
    3. **Object of class B is passed through either constructor or setter method - DI**
    4. Should be lightly coupled
18. Is-a
    1. Extends is a keyword used to implement inheritance
    2. Java supports following type
       1. Single
       2. Multilevel
       3. Hierarchical
    3. It does not support
       1. Multiple
       2. Hybrid
    4. If parent class has no constructors then a default implicit is added by default
    5. If parent class has even 1 parameterized constructor then no default constructor is added. Hence if there is even 1 parameterized constructor then
       1. Either do add a default constructor so child classes can easily inherit
       2. Or make sure to use super to call the **immediate** parent class constructor
    6. Creating a child class object invokes the topmost parent class constructor first, then the respective child classes in the multilevel hierarchy
19. Annotation : @Override, @Deprecated, @SuppressWarnings  
    they are small piece of information(meta data ) for the compiler or the JVM to get information about the location where it is used.
20. Overriding
    1. It can occur only in inheritance and not in same class
    2. Java uses @Override annotation to mark it as a method inherited from the parent class
    3. Rules
       1. Cannot change the access specifier, return type, method name or the list of parameters
       2. Can change the functionality of the method
21. Super
    1. It is a keyword
    2. Used from within a child class constructor to call the immediate parent class constructor and it has to be the 1st line in the constructor method
    3. It can also be used to call the parent class methods using . operator
22. Abstract classes
    1. abstract is a keyword
    2. it can be declared at class and method level only
    3. Abstract classes cannot be instantiated means cannot create objects using new
    4. Abstract classes may or may not have abstract methods and they can have constructors and other implemented methods as well
    5. If a class has even 1 abstract method then it is mandatory to declare the class itself as abstract
    6. Abstract methods are the methods that have no implementation or no body and they should either be implemented by the respective child classes or declare the child class itself as abstract
23. Interface
    1. interface is a keyword
    2. interfaces are by default abstract hence cannot be instantiated
    3. interface methods are by default abstract and public
    4. variables are by default public final and static
    5. interface can extend other interface
    6. classes implements interfaces and either provide the implementation of abstract methods or declare itself as abstract
    7. Any common functionalities across the classes irrespective of the relationship use interface
24. Reference decides what methods can be called. [ If it is parent class or interface reference then methods of only parent class or interface can be invoked]. But the method of which class will be called is decided at runtime based on the object passed at runtime  
    // item -> reference variable [ all methods of Item class ]  
    // methods of Book class will be called

Item item = new Book();

1. Annotations : small piece of information for the compiler or the JVM that provides extra information about the java components depending on where it is used.  
   Ex: @Override, @Deprecated
2. Exception Handling
   1. It is used to handle the runtime errors and for smooth functioning of the application
   2. There are 2 types of exceptions
      1. Checked – are checked by the compiler and reinforced to be handled
      2. Unchecked – are not checked by compiler and they are thrown at runtime
   3. Try – is wrapped around a block of code that may throw an exception
   4. Catch – is used to handle the exception
   5. Throw – is used to manually throw an exception
   6. Throws – is used at method level to let the user of method know that method throws an exception to be handled
   7. Finally – is a block that executes irrespective whether there is exception or not
   8. User defined exception -> using inheritance
3. Collections
   1. Provides advantage in terms of dynamic size
   2. Types of collection – ordered - unordered , sorted - unsorted
      1. List – ordered and unsorted + allows duplicates  
         ArrayList ( faster for iterations), LinkedList (modify ),   
         Vector is thread safe( slow – synchronized )
      2. Set – unordered and unsorted, no duplicates  
         hashSet LinkedHashSet  
         SortedSet interface extends Set - where by default the elements are arranged in asc order  
         TreeSet
      3. Map – unordered and unsorted, no duplicate key but allows duplicate values  
         HashMap(faster), Hashtable(slow – synchronized)  
         SortedMap – extends Map - by default keys are sorted in asc order  
         TreeMap
   3. Sorting
      1. Comparable => used directly on the class whose objects needed to be compared and provides a default implementation of sorting
      2. Comparator => to change the default sort and provide with custom sorting
   4. Anonymous inner class => a class that has no name but provides the implementation of the unimplemented method either in the interface or abstract class
   5. Collection of set => equals and hashcode
      1. Equals => checks for content
      2. Hashcode => if 2 objects land in same bucket or not
4. Data Structures
   1. Searching algorithms
      1. Linear
      2. Binary
   2. Sorting Algorithms
      1. Insertion
      2. Selection
      3. Bubble
      4. Quick
      5. Heap
      6. Tree sort
      7. Merge
   3. Stack => elements are always accessed from the top
      1. LIFO/ FILO
      2. Push
      3. Pop
      4. Overflow
      5. Underflow
      6. Top
   4. Queue =>
      1. FIFO
      2. Enqueue (push)
      3. Deque (pop)
   5. Binary Tree => There is always a root element and to the left of root element lies all the elements smaller to root and to the right all the elements greater than root
   6. LinkedList => basically links elements by storing the address of next element