|  |  |
| --- | --- |
| * **COMPARABLE and COMPARATOR both are INTERFACES and can be used to sort collection elements. Collection.sort(List) and Collection.sort(list, comparator).** | |
| * **ITERATOR is an INTERFACE to iterate over a collection. If a user is working with a for loop, they cannot modernize(add/remove) the Collection, whereas, if they use the Java Iterator, they can simply update the Collection.** | |
| * GENERICS is **a set of related methods or a set of similar types**. Generics allow types Integer, String, or even user-defined types to be passed as a parameter to classes, methods, or interfaces. Generics are mostly used by classes like HashSet or HashMap. | |
| * PROPERTIES CLASS **provides methods to get data from the properties file and store data into the properties file. Create db.properties file and create another class where we create FileReader and Properties Class object and pass db.pro** | |
| TRANSIENT is a variables modifier used in serialization. At the time of serialization, **if we don't want to save value of a particular variable in a file**, then we use transient keyword. When JVM comes across transient keyword, it ignores original value of the variable and save default value of that variable data type. | |
| An ENUM can, just like a CLASS, have attributes and methods. **The only difference is that enum constants are public, static and final (unchangeable - cannot be overridden).**  An ENUM cannot be used to create objects, and it cannot extend other classes (but it can implement interfaces). **ENUM can have fields, constructors and methods.** | |
| STATIC BLOCK: A class has to be loaded in main memory before we start using it. Static block is executed during class loading. | |
| CLASSLOADER is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the CLASSLOADER. | |
| The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer. | |
| IS-A relationship is inheritance. This means, that the child class is a type of parent class. For example, an apple is a fruit. So, you will extend fruit to get apple., HAS-A relationship is composition. composition means creating instances which have references to other objects. For example, a room has a table. So, you will create a class room and then in that class create an instance of type table. | |
| AGGREGATION (weak-Has-A) both objects can exist independently. COMPOSITION (strong) both objects cannot exist independently. One object cannot exist without owner object. | |
| If you want a class object to be serializable, all you need to do it implement the java.io.Serializable interface. Serialization in java is implemented by ObjectInputStream and ObjectOutputStream. | |
| char ch =s.charAt(2); charAt() returns a char value at the given index number index number start from 0 and goes to n-1. n is the length of the string. | |
| The String class valueOf() method converts given type such as int, long, float, double, boolean, char and char array into String. int a =10;  String s4 = String.valueOf(a); char ch1= 'A'; String s4=String.valueOf(ch1);  char vowel[] = {'A', 'E', 'I', 'O', 'U'}; String str = String.valueOf(vowel); | |
| String s = "Welcome to Coding World"; char[] ch= s.toCharArray(); | |
| String s1= "javatpoint"; String s2= s1.intern(); Heap Object to String Pool area  String s3= new String("javatpoint"); String s4= s3.intern(); | |
| The java string indexOf() method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.  String s1= "this is index of example"; int index1=s1.indexOf("is”); | |
| The Java String compareTo() method is used for comparing two strings lexicographically. If both the strings are equal then this method returns 0 else it returns positive or negative value. | |
| String str1 = "javatpointtt"; String[] arr = str1.split("t", 0); for (String w : arr) {  System.out.println(w);} split against delimiter/regular expression. | |
| //primitive to object --> AUTO BOXING  int x = 30; Integer y = Integer.valueOf(x);  //object into primitive ---UNBOXING  Double d = new Double(12.10);  //double d1 = d.doubleValue();  double d1 = d; | String x = "100";  int y = Integer.parseInt(x);  Integer m = 100;  int n = m.intValue();  double x = 10.3; CASTING  int y = (int) x; |
| String [] arr = {"10", "20"};  int p = Integer.parseInt(arr[0]);  int q = Integer.parseInt(arr[1]);  int r = p+q; | SWITCH statement executes one statement from multiple conditions.  char ch = 'O’; switch(ch) {  case 'a':  System.out.println("Vowel"); break; |
|  | |