* **JAVA ARCHITECTURE:**
* Source –file: This is the file that is saved as .java. All the source code is written in this file.
* For compilation we use the ‘javac’ compiler. One compilation we get a ‘.class’ file. This ‘.class’ file is understood by all operating systems.
* After this the file is run using ‘java’. This is run in ‘Java Virtual Machine’(JVM).
* JVM:
  + Loads the classes.
  + Verifies.
  + Converts to native code.
  + Executes main() statically.
  + Jdk 1.0 – java.
  + Jdk 1.2 to jdk 1.5 – Java 2 .
  + Jdk 1.5 -1.7 – Java 5. (Smart – Lazy Loading).
* **JAVA DATATYPES:**
* Byte :
  + 1 byte
  + +127 to -127.
* boolean:
  + 1 byte
  + true or false(default).
  + Only data type that can be used as a standalone in an expression.
* char:
  + 2 bytes.
  + 0-65535 Unicode Characters of which first 0-255 are ASCII.
* short:
  + 2 bytes.
  + +32767 to -32767.
* int:
  + 4 bytes.
  + +2147483647 to -2147483647.

long:

* + 8 bytes
  + +9223372036854775007 to -9223372036854775007.
* float:
  + 4 bytes.
  + +3.4 \*10^38 to -3.4 \*10^38.
* Double:
  + 8 bytes.
  + +1.7\*10^308 to -1.7\*10^308.

1. All the above data types have Wrapper classes like int has Integer which convert the data types from primitives to objects.
2. All arithmetic, logical and relational operators only work with primitives and not with objects.

Int x,y,z;

Integer n;

x = 10;

y = 5;

* n = new Integer(100); //wrapping.
* Z = x+ y +n; //Wont work because n is not a primitive but an object.
* The above didn’t work till jdk 1.5 After 1.5 due to smart compilation this would work as n = 100 will be converted to n = new Integer(100) while compiling. Similar thing will be done for z = x+y+n.
* **STRING IN JAVA:**
* These are objects of a class String or Double.gnote constants.
* Strings in java are immutable. Meaning if a string is formed it remains in the string pool of JVM. If any new operation is performed a new string is formed but the original string is not modified.

int n;

char ch;

String s1,s2;

s1 = “Hamburger”;

s3 = “ to be or not to be”;

n = s1.length(); //This proves that TOM is an instance.

ch = s1.charAt(0); //o/p = H.

s2 = s1.substring(3); // o/p = burger.

s2 = s1.substring(4,8); // o/p = urge.

n = s3.indexOf(“and”); // o/p = -1.

n = s3.indexOf(“not”); // o/p = 9.

While(n!=-1)

{

n = s3.indexOf(“to”, n+1) //looping this in while will give all occurrences

} // of the string.

if(s1.equals(“Hamburger”)) //To check whether a given string is as

{ // expected.

* system.out.println(“The string is Hamburger”)
* }
* if(s1.equalsIgnoreCase(“hamburger”)) //To check whether a given string
* { // is as expected ignoring cases.
* system.out.println(“The string is Hamburger”)
* }
* if(s1.startsWith(“Ha”)) {}
* if(s1.endsWith(“ger”)) {}
* s1 = s1.toUpperCase()
* s1 = s1.toLowerCase()
* s1.trim(); //Removes leading and trailing spaces.

s1.compareTo(s2) // if true returns 0, if s1>s2 +ve, s1<s2 –ve, with

* // return type int.
* In the above program if(s1==”Hamburger”) should not be used as s1 is object of class string.

s1 = s1 + “ is good” //s1 = Hamburger is good. String Concatenation.

n=100;

s1 = s1 + n; //s1 = Hamburger100.