What is the difference between JDK and JRE?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The **JDK (Java Development Kit)** and **JRE (Java Runtime Environment)** are both essential components of Java, but they serve different purposes:  **1. JDK (Java Development Kit)**   * **Purpose:** Used for developing, compiling, and running Java applications. * **Includes:**   + JRE (Java Runtime Environment)   + Java Compiler (javac)   + Development tools (debugger, Javadoc, etc.) * **Who Needs It?** Developers who write and compile Java programs.   **2. JRE (Java Runtime Environment)**   * **Purpose:** Used only for running Java applications. * **Includes:**   + JVM (Java Virtual Machine)   + Java class libraries * **Who Needs It?** End-users who just want to run Java applications but don’t need to develop them.   **Key Difference:**   |  |  |  | | --- | --- | --- | | **Feature** | **JDK (Java Development Kit)** | **JRE (Java Runtime Environment)** | | **Includes JVM?** | ✅ Yes | ✅ Yes | | **Includes JRE?** | ✅ Yes | ❌ No (JRE is a standalone package) | | **Includes Compiler?** | ✅ Yes (javac) | ❌ No | | **Usage** | Development & Execution | Execution only |   **In short:**   * If you want to **develop Java applications**, install the **JDK**. * If you only need to **run Java applications**, install the **JRE**.   Difference Between JDK, JRE, and JVM | Board Infinity  What jdk(Java development kit) contains?  It contains set of tools and JRE  What is JRE(Java Runtime Environment)?  It contains JVM + JCL  What is JVM?  Java Virtual Machine which is used to run the java programs  What is JCL (Java Class Library)  Set of pre-defined classes. |

What is a package?

|  |
| --- |
| 1. A package is a container of classes. 2. A package contains collection classes. 3. A package is a folder which contains classes, interfaces, enums and annotations.   There are 2 types of packages   1. Pre-defined(provided by inventors of Java) 2. User-defined(Written by programmers)   Some of the pre-defined available in Java are:   1. Java.lang 2. Java.util 3. Java.io 4. Java.awt 5. Java.text 6. Java.awt.event 7. Java.sql 8. Java.net 9. Java.util.stream 10. Etc.. |

What is java API (Application Programming Interface)?

|  |
| --- |
| Collection of pre-defined packages |

We can write comments in 3 ways?

|  |
| --- |
| 1. Single line (//………………) 2. Multi line comments (/\*……….\*/) 3. Documentation comments (/\*\*…………………………………\*/) |

What is the use of “import java.lang.\*” statement in our program?

|  |
| --- |
| It is a statement which is used to import a package. |

What is Java API?

|  |
| --- |
| Collection of pre-defined packages |

What is JCL(Java Class Library)?

|  |
| --- |
| Collection of pre-defined classes. |

What is a keyword?

|  |
| --- |
| It is a pre-defined word, provided by inventors of Java |

What is a class?

|  |
| --- |
| A class is a block which contains functions(methods) and variables(fields/attributes/properties) |

**Sixth.java**

|  |
| --- |
| //this is my sixth program  /\* multi line comments  Author: Balaji B  Date: 08-Feb-2025  \*/  /\*\* documentation comments  Org: Madhu Tech Skills  City: vijayawada  \*/  package p1; //creating a package by using a keyword called package  import java.lang.\*; //importing a package called java.lang  interface MyInterface{}  enum Colors{  }  @interface MyAnnotation{}  class Ayyo{  public static void main(String[] args)  {  System.out.println("Ayyaaaa...");  }  } |

**What is package p1; statement?**

|  |
| --- |
| * It is a statement which is used to create a new package. Here package is a keyword and p1 is the package name. |

**What is the use of “import java.lang.\*;” ?**

|  |
| --- |
| It is used to import all the classes, interfaces, enums and annotations existed in a package called java.lang. |

**Why we import classes of a package?**

|  |
| --- |
| To use those classes, you have to import them |

**Can we write print statement outside the function in Java?**

|  |
| --- |
| No we can’t, but we can write it in any block which is written in a class. |

**If you write print statement outside the function what error you will get?**

|  |
| --- |
|  |

**Can I declare a variable with in a class and outside the function?**

|  |
| --- |
| **Yes we can declare 2 types of variables**   * Instance variables (it is not declared as static) * Static variables (it is declared as static) |

**Can I write statements other than declarations in a class and outside the functions?**

|  |
| --- |
| No we can write only variable declaration(definition) statements. |

**If I write the “c=a\*s;” statement outside the function within a class what kind of error you will get?**

|  |
| --- |
|  |

**What is String, System which are used in our program?**

|  |
| --- |
| * These are a pre-defined classes existed in java.lang package |

**Can use String class without importing java.lang package?**

|  |
| --- |
| Yes, we can because java.lang package is the default package, which means the compiler imports it automatically. |

**When the java.lang package will be imported automatically?**

|  |
| --- |
| During compilation |

What is a string?

|  |
| --- |
| A string is nothing but collection of characters represented by pair of double quotations  Ex-1: “ravi teja” -> string literal(value)  Ex-2: “Vamsi” -> string literal  Ex-3: “12345” -> string literal  Ex-4: “+-\*&^@” -> string literal |

What is a concatenation symbol?

|  |
| --- |
| If you find a ‘+’ symbol after or before a string literal it is called as a concatenation symbol |

What is the use of concatenation symbol?

|  |
| --- |
| It appends any value to the string  Ex: “a=”+10 result is “a=10” |

What are the differences between print and println()?

|  |  |
| --- | --- |
| Print | Println |
| 1. It prints the data without moving cursor to the new line | 1. It moves the cursor to the new line after printing the data. |
| 1. We can pass only one argument to the print function | 1. We can pass zero or one argument to the println function |

Third.java

|  |
| --- |
| //this is my sixth program  /\* multi line comments  Author: Balaji B  Date: 08-Feb-2025  \*/  /\*\* documentation comments  Org: Madhu Tech Skills  City: vijayawada  \*/  package p1; //creating a package by using a keyword called package  import java.lang.\*; //importing a package called java.lang  interface MyInterface{}  enum Colors{  }  @interface MyAnnotation{}  class Ayyo  {  static int s=1;  int a=2;  int c=a+s;  //c=a\*s; Error non declaration statements are not allowed here  public static void main(String[] args)  {  int a=2,b=3,c=a+b;  System.out.println("Ayyaaaa...");  System.out.println(); //it displays new line  System.out.println(a,b,c); //error  System.out.print(); //error  }  }  //this is my sixth program  /\* multi line comments  Author: Balaji B  Date: 08-Feb-2025  \*/  /\*\* documentation comments  Org: Madhu Tech Skills  City: vijayawada  \*/  package p1; //creating a package by using a keyword called package  import java.lang.\*; //importing a package called java.lang  interface MyInterface{}  enum Colors{  }  @interface MyAnnotation{}  class Ayyo  {  static int s=1;  int a=2;  int c=a+s;  //c=a\*s; Error non declaration statements are not allowed here  public static void main(String[] args)  {  int a=2,b=3,c=a+b;  System.out.println("Ayyaaaa...");  System.out.println(); //it displays new line  //System.out.println(a,b,c); //error  //System.out.print(); //error  }  }  How to compile the program to create a package?     * + 1. -d: it tells the compiler to create a folder with package name     2. . : it tells where to create the folder (Here . means present folder)   If you compile the above program like below then the package will be create din hari3 folder  Compile: javac -d C:\Users\kalla\OneDrive\Dokumen\hari3 Sixth.java |

How to declare a variable?

|  |
| --- |
| In Java we can declare a variable by using a data type  Syntax: <data-type> <var-name>[=value]; Ex: int a; Ex: float f=10.00f;  Syntax: <data-type> <var-1[=value]>[,var-2[=value],var-3[=value]…var-n[=value]]; |

Example to declare multiple variables

Four.java (no need to execute this program)

|  |
| --- |
| import java.lang.\*; //import statement  class Four  {  public static void main(String args[])  {  int a,b,c; //multiple variables declaration of same type  int x=10,y=20,z=0;// defining varaibles.  byte b;  short s;  int i;  long l;  //we can't declare multiple types of variables in a single statement;  }  } |

What is a local variable?

|  |
| --- |
| 1. It is a variable which is declared within a block existed in a class is called as local variable 2. Local variables are created either in java stack memory or native method stacks memory. 3. Local variables will not be initialized with default values, so you have to initialize them before usage. |

What error you will get if we use local variable without initializing it?

|  |
| --- |
| **Seven.java**  public class Seven {      public static void main(String[] args) {          int a,b,c; //3 local variables          System.out.println(a);      }  }  Compilation error:  Seven.java:4: error: variable a might not have been initialized  System.out.println(a);  ^  1 error |

Why java is called as extensible programming language?

|  |
| --- |
| Because in Java we can use native methods |

What are native methods?

|  |
| --- |
| 1. Methods which are written in other languages like C are called as native methods 2. It is possible to use native methods in Java   Ex: public static native long currentTimeMillis(); method of System class is a native method |

**Formatting methods**

**What are formatting () methods:**

In jdk1.5 version two new methods are introduced, to replace the print() and println() methods, and these methods are called as formatting methods.

**Note:** these methods are called as formatting methods because these methods formats the given string by using converters and then prints the formatted string.

What is the first argument we should pass to the formatting method?

|  |
| --- |
| String literal |

Those are:

* 1. Printf()
  2. Format()

These methods are existed in the java.io.PrintStream class and these methods, format and printf, are equivalent to one another.

**What is the difference between printf() and format() method?**

|  |
| --- |
| Actually **printf()** method is internally calls the **format()** method, that is actual code is existed in the format method. The printf() method is just provided for our convenience. |

**Some of the converters which are used in format method**

|  |  |  |
| --- | --- | --- |
| Converter | Flag | Explanation |
| d |  | A decimal integer |
| f |  | A float |
| n |  | A new line character appropriate to the platform running the application. You shold always use %n rather than \n |
| tB |  | Full name of month |
| tb |  | Short name of month |
| td |  | day of month, if it is single digit, it puts zero before that number |
| te |  | Day of month, it will not put zero before single digit. |
| ty |  | 2-digit year |
| tY |  | 4 digit year |
| tl |  | Hour in 12 hours clock |
| tM |  | Minutes in 2 digits, with leading zeros as necessary |
| tS |  | Seconds |
| tp |  | Am/pm |
| tm |  | Month in 2 digits, with leading zeros as necessary |
| tD |  | Date as %tm%td%ty |
|  | 08 | 8 characters in width, with leading zeros as necessary |
|  | + | Includes sign, whether positive or negative |
|  | , | Includes Locale-specific grouping characters. |
|  | - | Left-justified |
|  | .3 | Three places after decimal part |
|  | 10.3 | Ten characters in width, right justified, with three places, after decimal point |

**Printf():**

It is existed in the PrintStream class; it takes 1 or more arguments.

**FormattingMethods1.java**

|  |
| --- |
| import java.util.Date;  public class Seven {      public static void main(String[] args) {          int a,b,c; //3 local variables          a=10;          b=3;          c=a-b;          System.out.format("Hello%n");          System.out.format("%d-%d=%d%n",a,b,c);          System.out.println(a+"-"+b+"="+c);            //Native method usage          long l=System.currentTimeMillis();          System.out.println("l:\t"+new Date(l));      }  }  **Output:**  Hello  10-3=7  10-3=7  l: Sat Feb 08 16:26:14 IST 2025 |

**FormattingMethodsDemo**

|  |
| --- |
| import java.util.Date;  public class Seven  {      public static void main(String[] args)      {          String name="madhu";          int i=100;          float f=200;          double d=200;          char ch='A';          boolean bl=true;          System.out.printf("name:\t%s%n",name);          System.out.printf("i:\t%d%n",i);          System.out.printf("f:\t%.2f%n",f);          System.out.printf("d:\t%.3f%n",d);          System.out.printf("ch:\t%c%n",ch);          System.out.printf("bl:\t%s%n",bl);      }  }  Output:  name: madhu  i: 100  f: 200.00  d: 200.000  ch: A  bl: true |

**%s can be used for any type of object or value(see the below example)**

|  |
| --- |
| import java.util.Date;  public class Seven  {      public static void main(String[] args)      {          String name="madhu";          int i=100;          float f=200;          double d=200;          char ch='A';          boolean bl=true;          System.out.printf("name:\t%s%n",name);          System.out.printf("i:\t%s%n",i);          System.out.printf("f:\t%s%n",f);          System.out.printf("d:\t%s%n",d);          System.out.printf("ch:\t%s%n",ch);          System.out.printf("bl:\t%s%n",bl);      }  }  **Output:**  name: madhu  i: 100  f: 200.0  d: 200.0  ch: A  bl: true |

**FormatDemo3.java**

|  |
| --- |
| import static java.lang.System.\*;  class FormatDemo3  { public static void main(String args[])  { long n = 261011;  out.format("%d%n",n); // --> "261011"  out.format("%09d%n",n); // --> "000261011"  out.format("%9d%n",n); // --> " 261011"  out.format("%,9d%n",n); // --> " 261,011"  out.format("%+,9d%n%n",n); // --> " +261,011"  }  }  Output:  261011  000261011  261011  261,011  +261,011 |

**FormatDemo4.java(Example on alignment)**

|  |
| --- |
| public class Eight {      public static void main(String[] args) {          double pi =3.141593;          System.out.format("%f%n", pi);       // -->  "3.141593"          System.out.format("%.3f%n", pi);     // -->  "3.142"          System.out.format("%10.3f%n", pi);   // -->  "     3.142"          System.out.format("%-10.3f%n", pi);  // -->  "3.142     "// left justified          System.out.printf("%3f%n",pi);  //it displays total value          System.out.printf("Madhu Tech Skills...");      }  }  **Output:**  3.141593  3.142  3.142  3.142  3.141593  Madhu Tech Skills... |

**FormatDemo5.java**

|  |
| --- |
| import static java.lang.System.\*;  import java.util.\*;  class FormatDemo5  { public static void main(String args[])  { Date date=new Date();  out.println("Actual Date:\t"+date);  out.printf("Month=%tB %n",date);  out.printf("Day=%td%n",date);  out.printf("Year=%tY%n",date);  Calendar c = Calendar.getInstance();  out.format("%tB %td, %tY%n", c, c, c); // --> "July 09, 2013"  out.format("%tb %te, %tY%n", c, c, c); // --> "Jul 9, 2013"  out.format("%tl:%tM %tp%n", c, c, c); // --> "2:34 am"  out.format("%tD%n", c); // --> "07/09/13"//month/day/year  }  } |

**FormatDemo6.java**

|  |
| --- |
| import static java.lang.System.\*;  import java.util.\*;  class FormatDemo6  { public static void main(String args[])  { Date sd=new Date();  out.println("SystemDate:\t"+sd);  out.format("%td-%tB-%tY%n",sd,sd,sd);  out.format("%te-%tb-%ty%n",sd,sd,sd);  out.printf("%tl:%tM:%tS %tp",sd,sd,sd,sd);  }  } |

**Functions**

**What is a function?**

|  |
| --- |
| * A function is a block which contains re-usable set of statements * We write a function to perform a task * Once the function is written, to execute the statements of it, we have to call it |

A function is a block which has a name ends with pair of parentheses and the name is preceded by a data type (it is called as return type). It contains set of statements which are used to perform a task.

What is a static method?

|  |
| --- |
| * 1. If a method is declared by using static keyword then it is called as static method   2. We can call a static method in another static method of same class directly.   Note: we can’t call non-static method (instance method) in a static method directly. Even though it is existed in the same class. |

Who calls the main function?

|  |
| --- |
| JVM calls the main function |

What is the advantage of functions?

|  |
| --- |
| Functions provide several advantages in programming:   1. **Code Reusability** – Functions allow you to reuse code multiple times without rewriting it, reducing redundancy. 2. **Modularity** – They break a large program into smaller, manageable parts, making the code easier to understand and maintain. 3. **Improved Readability** – Well-named functions make the code more readable and self-explanatory. 4. **Easier Debugging** – Errors are easier to find and fix when the code is modular, as you can test individual functions separately. 5. **Scalability** – Functions help in building scalable applications by organizing code efficiently. 6. **Encapsulation** – They allow you to hide implementation details and expose only necessary functionality, improving security and organization. 7. **Avoiding Code Duplication** – Instead of writing the same logic multiple times, you can call a function whenever needed. 8. **Efficient Memory Use** – Functions help optimize memory usage by executing only when called and releasing resources when done. |

**A simple example on functions**

|  |
| --- |
| public class Nine  {      static void add(){          int a=10,b=3,c;          c=a+b;          System.out.printf("%d + %d = %d %n",a,b,c);      }      public static void main(String[] args) {          add();      }  }  **Output:**  10 + 3 = 13 |

**Another example on functions with parameters**

|  |
| --- |
| public class Nine  {      static void add()      {          int a=10,b=3,c;          c=a+b;          System.out.printf("%d + %d = %d %n",a,b,c);      }      static void sub(int a,int b){          int c;          c=a-b;          System.out.printf("%d - %d = %d %n",a,b,c);      }      public static void main(String[] args) {          add();          sub(100,200);      }  }  **Output:**  10 + 3 = 13  100 - 200 = -100 |

**What is a parameter?**

|  |
| --- |
| * + 1. Variables which are declared within parenthesis of a function are called as parameters.     2. Parameters are separated by comma (,)     3. After last parameter no need to give semicolon or comma |

**Example on Types of functions**

|  |
| --- |
| public class Nine  {      //function without parameters and without return type      static void add()      {          int a=10,b=3,c;          c=a+b;          System.out.printf("%d + %d = %d %n",a,b,c);      }      //function with parameters and without return type      static void sub(int a,int b){          int c;          c=a-b;          System.out.printf("%d - %d = %d %n",a,b,c);      }      //function with parameters and with return type      static int multi(int a,int b)      {          int c=a\*b;          return c;      }      //function without parameters and with return type      static int div()      {          int a=10,b=3;          return a/b;      }      public static void main(String[] args) {          add();          sub(100,200);   //to the sub function i am passing two arguments          int x=multi(10, 5); //to the multi() function i am passing two arguments and it is returning the result(int)          System.out.println("x:\t"+x);          int r1=x/2;          System.out.println("r1:\t"+r1);          System.out.println( div() );      }  }  **Output:**  10 + 3 = 13  100 - 200 = -100  x: 50  r1: 25  3 |

**Variable length arguments**

**Example on variable length argument**

|  |
| --- |
| public class Ten  {      static void display(int... enos)      {          System.out.println("..................");          for(int n:enos)          System.out.println(n);      }      public static void main(String[] args)      {          display();          display(1);          display(1,2);          display(1,2,3);      }  }  Output:  ..................  ..................  1  ..................  1  2  ..................  1  2  3 |

**Note:** A method can have variable length parameters with other parameters too, but one should ensure that there exists only one varargs parameter that should be written last in the parameter list of the method declaration. For example:

Another example on variable length arguments

|  |
| --- |
| public class Ten  {      static void display(String ename, int eno,int... marks)      {          System.out.printf(".....%s..with....eno....%d...marks....%n",ename,eno);          for(int mark:marks)          System.out.println(mark);      }      public static void main(String[] args)      {          display("Madhu.K",101,78,77,66,77,88);          display("Manish",102,78,77,66);          display("Lakshman",103,78,77,66);        }  }  Output:  .....Madhu.K..with....eno....101...marks....  78  77  66  77  88  .....Manish..with....eno....102...marks....  78  77  66  .....Lakshman..with....eno....103...marks....  78  77  66 |

What is a method?

|  |
| --- |
| A method is a function written in a class. |

What is the starting point of the program?

|  |
| --- |
| 1. Main method 2. In Java main method should be a static method 3. In Java main method return type should be only void.   Note: In Java you should declare the main method as public and static |

Reading data from keyboard by using readLine() method of DataInputStream class?

|  |
| --- |
| * readLine() method is a non-static method existed in DataInputStream class, so we have to call it by using object of DataInputStream class * readLine() method returns string only.. |

How to read the data from keyboard?

|  |
| --- |
| By using readLine () method we can read the data see the below example  import java.io.DataInputStream;  import java.io.IOException;  public class Eleven {      public static void main(String[] args) throws IOException      {          //Creating object for DataInputStream class          DataInputStream dis=new DataInputStream(System.in);          System.out.print("Enter your name:\t");          String name=dis.readLine();  //  name <-  = <- "sambasivarao" <- readLine()  <- sambasivarao <- keyboard          System.out.println("Hi "+name);        }  }  Output: |

**Operators**

What is an operator?

|  |
| --- |
| Operator is a symbol, which is used to perform an operation.  **Different types of operators supported by Java**  Ex: +, -, \*, /, %, (arithmetic operators)  Ex: <,>,<=,>=,==,!=,instanceof (relational operators  Ex: &&,||, ! (logical operators)  Ex: &,|,^,<<,>>, ~, >>>(Bitwise Operators)(zero fill right shift operator)  Ex: = (assignment operator)  Ex: new (memory allocation operator), . (dot is called as memory access operator), () type cast operator  Ex: ++,--, +=,-=,\*=,/=,%= (short cut operators) |

Example on Arithmetic Operators

|  |
| --- |
| import java.io.DataInputStream;  import java.io.IOException;  public class Eleven {      public static void main(String[] args) throws IOException      {          DataInputStream dis=new DataInputStream(System.in);          System.out.print("Enter int:\t");          String s1=dis.readLine();  // a <-  = <- "10" <- readLine()  <- 10 <- keyboard          int a=Integer.parseInt(s1);          System.err.print("Enter another int value:\t");          String s2=dis.readLine();          int b=Integer.parseInt(s2);          System.out.println(a+b);          System.out.println(a-b);          System.out.println(a\*b);          System.out.println(a/b);          System.out.println(a%b);      }  }  Output:  Enter int: 10  Enter another int value: 2  12  8  20  5  0 |

What is the drawback of readLine() of DataInputStream class? And why it is deprecated?

|  |
| --- |
| * by using this method, we can read only the characters which are in a range of 0 to 255 (ASCII character set). * this method can’t read other than ASCII character set (i.e. it can’t read all the characters of Unicode character set). * The method has been deprecated since Java 1.1 because it does not properly convert bytes to characters in a way that supports internationalization. * readLine() reads input as a **byte stream** and does not properly decode characters according to the character encoding (e.g., UTF-8). This can lead to **incorrect character conversions**, especially for non-ASCII characters. |

readLine() method of BufferedReader class

|  |
| --- |
| * 1. by using this method, we can read the characters which are in Unicode character set. |

Example on usage of readLine() method of BufferedReader class

|  |
| --- |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class Eleven {      public static void main(String[] args) throws IOException      {          InputStreamReader isr=new InputStreamReader(System.in);          BufferedReader br=new BufferedReader(isr);          System.out.print("Enter int:\t");          String s1=br.readLine();  // a <-  = <- "10" <- readLine()  <- 10 <- keyboard          int a=Integer.parseInt(s1);          System.err.print("Enter another int value:\t");          String s2=br.readLine();          int b=Integer.parseInt(s2);          System.out.format("%d + %d = %d%n",a,b,a+b);          System.out.format("%d - %d = %d%n",a,b,a-b);          System.out.format("%d \* %d = %d%n",a,b,a\*b);          System.out.format("%d / %d = %d%n",a,b,a/b);          System.out.format("%d %% %d = %d%n",a,b,a%b);        }  }  **Output:**  Enter int: 12  Enter another int value: 3  12 + 3 = 15  12 - 3 = 9  12 \* 3 = 36  12 / 3 = 4  12 % 3 = 0 |

Where the parseInt() method is existed?

|  |
| --- |
| * It is existed in Integer class * It takes string as an argument and converts it into int value and returns that int value |

Relational Operators

|  |
| --- |
| Relational Operators are used to compare 2 values(operands), after comparison these operators gives us a Boolean value(true/false) |

Example on relational operators

|  |
| --- |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class Twelve  {      public static void main(String[] args) throws IOException       {          InputStreamReader isr = new InputStreamReader(System.in);          BufferedReader br = new BufferedReader(isr);          System.out.print("Enter a number :\t");          int i = Integer.parseInt(br.readLine());          System.out.print("Enter another number :\t");          int j = Integer.parseInt(br.readLine());          System.out.println("i:\t "+i);          System.out.printf("j:\t%d %n",j);          // test case-1          //i=10          //j=5          boolean b1= i>j; //b1=true          boolean b2=i>=j; // (i>j or i==j)  b2=true          boolean b3=i<j;  //b3=false          boolean b4= i<=j; //i<j or i==j  b4=false          boolean b5=i==j; //b5=false          boolean b6=i!=j; //b6=true          System.out.println("i>j:\t"+b1);          System.out.println("i>=j:\t"+b2);          System.out.println("i<j:\t"+b3);          System.out.println("i<=j:\t"+b4);          System.out.println("i==j:\t"+b5);          System.out.println("i!=j:\t"+b6);      }  }  Output:  Enter a number : 10  Enter another number : 5  i: 10  j: 5  i>j: true  i>=j: true  i<j: false  i<=j: false  i==j: false  i!=j: true |

**What is an operand?**

|  |
| --- |
| It is a value on which the operator performs an operation.  Ex: 10+20 Here 10,20 are operands and + is an operator |

**Logical Operators (Boolean operators)**

|  |
| --- |
| * 1. Logical operators are used to perform operation on Boolean values only   2. These operators gives us Boolean values as a result. * Logical and operator (&&): it gives us true if both operands are true, otherwise false * Logical or operator (||):it gives us true if any one or both operands are true, otherwise false * Logical not operator (!): it returns true if operands is false vice versa. |

How parseBoolean() method works?

|  |
| --- |
| * 1. It converts “TRUE” to true and returns it   2. It converts “true” to true and returns it   3. If any string other than “true” (case-insensitive) is passed then It returns false |

Example on parseBoolean() method which returns true

|  |
| --- |
| public class Fourteen  {      public static void main(String[] args) {          System.err.println(Boolean.parseBoolean("true"));          System.err.println(Boolean.parseBoolean("truE"));          System.err.println(Boolean.parseBoolean("trUE"));          System.err.println(Boolean.parseBoolean("True"));          System.err.println(Boolean.parseBoolean("TRUE"));      }  }  Output:  true  true  true  true  true |

Example on parseBoolean() method which returns false

|  |
| --- |
| public class Fourteen  {      public static void main(String[] args) {          System.err.println(Boolean.parseBoolean("false"));          System.err.println(Boolean.parseBoolean("madhu"));          System.err.println(Boolean.parseBoolean("1234"));          System.err.println(Boolean.parseBoolean("Flaws"));          System.err.println(Boolean.parseBoolean("FALSE"));          System.err.println(Boolean.parseBoolean("TOKKALEY"));          System.err.println(Boolean.parseBoolean("1"));          System.err.println(Boolean.parseBoolean("C"));      }  }  Output:  false  false  false  false  false  false  false  false |

Example on logical operators

|  |
| --- |
| import java.io.BufferedReader;  import java.io.InputStreamReader;  public class Thirteen {      public static void main(String[] args) throws Exception      {          InputStreamReader isr=new InputStreamReader(System.in);          BufferedReader br=new BufferedReader(isr);          System.out.print("Enter a boolean value:\t");          boolean a = Boolean.parseBoolean(br.readLine());  //"true"          System.out.print("Enter another boolean value:\t");          boolean b = Boolean.parseBoolean(br.readLine());  //"true"          System.out.println("a:\t"+a);          System.out.println("b:\t"+b);          System.out.println("a&&b:\t"+(a&&b));          System.out.println("a||b:\t"+(a||b));          System.out.println("!a:\t"+!a);          System.out.println("!b:\t"+!b);        }  }  Output:  Enter a boolean value: madhu  Enter another boolean value: TrUe  a: false  b: true  a&&b: false  a||b: true  !a: true  !b: false |

How to read a single character (within a range of 0 to 255)?

|  |
| --- |
| public class Fifteen {      public static void main(String[] args) throws Exception      {          System.out.println("Enter any character:\t");          int n=System.in.read();  //it reads only single character within range of 0 t 255          System.out.println("n:\t"+n);      }  }  Output:  Enter any character:  A  n: 65 |

Is JVM able to display Telugu letter అ(see the below example)

|  |
| --- |
| import java.io.BufferedReader;  import java.io.InputStreamReader;  public class Fifteen {      public static void main(String[] args) throws Exception      {          char ch1='\u0C05'; //in ch1 i am storing telugu letter అ it's unicode value is 3077          System.out.println("int(ch1):\t"+(int)ch1);          System.out.println("int:\t"+ch1);      }  }  **Output:**  int(ch1): 3077  int: ? |

Can we declare a local variable as static?

|  |
| --- |
| No, local variables can’t be either static or instance |

What is a literal?

|  |
| --- |
| 1. It is a value used in a program   Different types of literals supported by Java   1. Integer literal (10,20) 2. Float literal (10.0,2.0) 3. String literal(“madhu”,”123”) 4. Boolean literal(true,false) 5. Character literal (‘a’,’1’) |

What is a token?

|  |
| --- |
| Each and every individual unit in a program is called as token.  The list of tokens supported by Java   1. Keywords 2. Identifiers 3. Literals 4. Operators 5. Separators 6. Special characters 7. Comments |

What is String in “String args[]”?

|  |
| --- |
| It is a pre-defined class existed in java.lang package |

What is a System in “System.out.println()”?

|  |
| --- |
| It is a pre-defined class existed in java.lang package |

What is a println() in “System.out.println()”?

|  |
| --- |
| It is a pre-defined method |

What is a reference variable?

|  |
| --- |
| If we declare a variable by using class name as a data type. Then those variables are called as reference variables (like pointer variables in C/C++). |

What is a reference data type?

|  |
| --- |
| It is a data type which is used to declare reference variables. |

Why we write a class?

|  |
| --- |
| 1. We write a class to create objects |

How many times the memory will be allocated for a static variable in a program?

|  |
| --- |
| 1. Only once |

How many times the memory will be allocated for instance variables in a program?

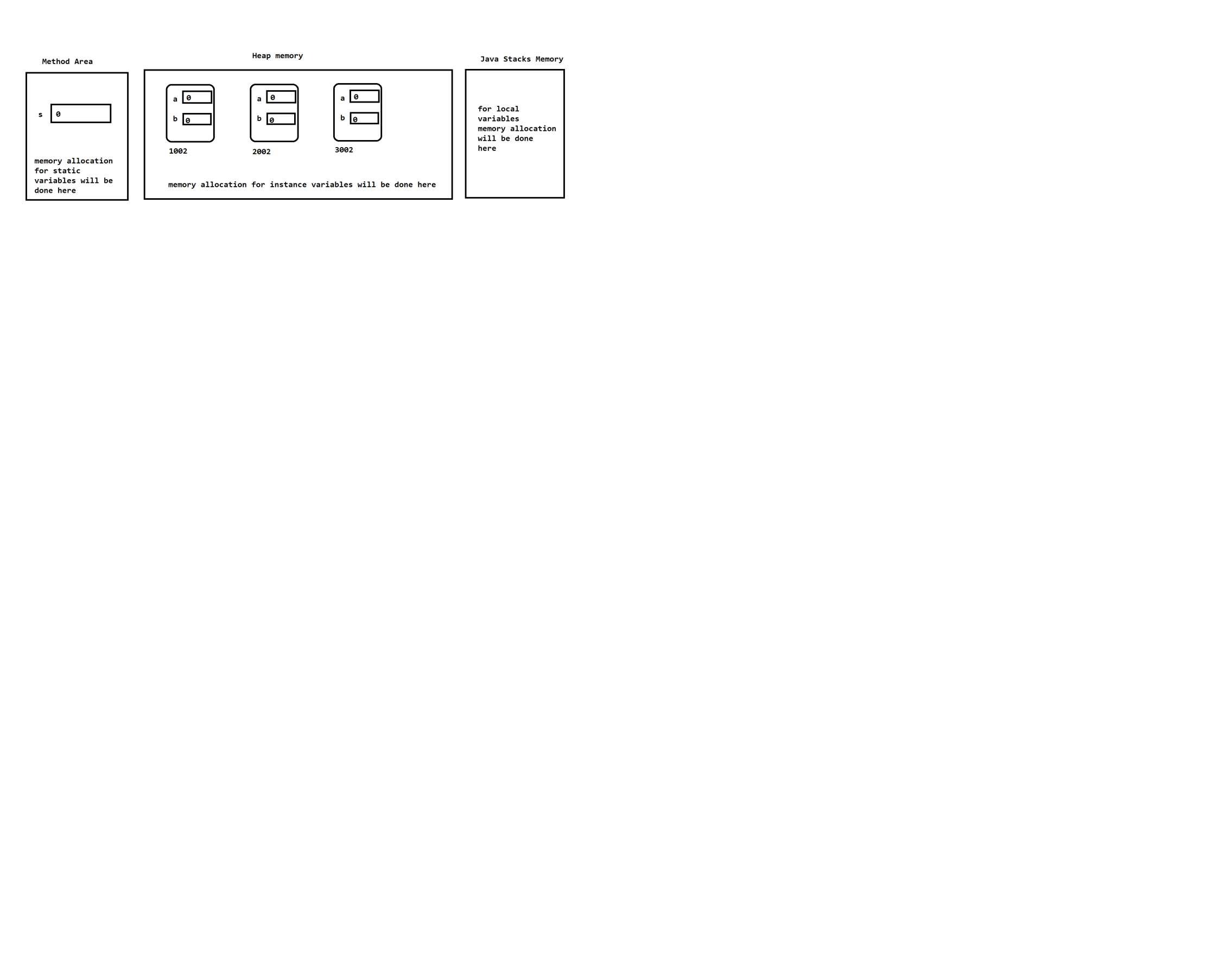
|  |
| --- |
| 1. N times if you create n objects. |

How to create an object?

|  |
| --- |
| We can create an object by using   1. new operator (new) 2. Constructor calling statement ( Dog()-> it is called as constructor calling ) |

Seven.java

|  |
| --- |
| import java.lang.\*; //import statement  class Dog  { static int s;  int a,b;  }  class Seven  { public static void main(String args[])  { int a;  Dog e1=new Dog();// object creation  Dog e2=new Dog();// object creation  Dog e3=new Dog();// object creation  }  } |



**Modified code (Seven.java)**

|  |
| --- |
| import java.lang.\*; //import statement  class Dog  {  static int s;  int a,b;  }  class Seven  {  public static void main(String args[])  {  int a;  //e1=1002  Dog e1=new Dog();// object creation  System.out.println("e1......object.....state....");  System.out.println(e1.a);  System.out.println(e1.b);  //e2=2002  Dog e2=new Dog();// object creation  System.out.println("e2......object.....state....");  System.out.println(e2.a);  System.out.println(e2.b);  //e3=3002  Dog e3=new Dog();// object creation  System.out.println("e3......object.....state....");  System.out.println(e3.a);  System.out.println(e3.b);  }  }  **Output:**  e1......object.....state....  0  0  e2......object.....state....  0  0  e3......object.....state....  0  0 |

What is object initialization?

|  |
| --- |
| Process of assigning values to the fields/variables existed in an object. |

Seven.java

|  |
| --- |
| import java.lang.\*; //import statement  class Dog  {  static int s;  int a,b;  }  class Seven  {  public static void main(String args[])  {  int a;  //e1=1002  Dog e1=new Dog();// object creation  //e1 object initialization  e1.a=100;  e1.b=200;  System.out.println("e1......object.....state....");  System.out.println(e1.a);  System.out.println(e1.b);  //e2=2002  Dog e2=new Dog();// object creation  //e2 object initialization  e2.a=300;  e2.b=400;  System.out.println("e2......object.....state....");  System.out.println(e2.a);  System.out.println(e2.b);  //e3=3002  Dog e3=new Dog();// object creation  //e3 object initialization  e3.a=500;  e3.b=600;  System.out.println("e3......object.....state....");  System.out.println(e3.a);  System.out.println(e3.b);  }  }  **Output:**  C:\Users\Madhu.K\Documents\JavaPrograms-1>java Seven  e1......object.....state....  100  200  e2......object.....state....  300  400  e3......object.....state....  500  600 |

What Is an instance method?

|  |
| --- |
| 1. It is a method written in a class and not defined as static. 2. We can call instance methods by using object   Ex: Dog d1=new Dog(); //object creation  Ex: d1.display(); //instance method calling |

How to access static variables?

|  |
| --- |
| We can access static variables by using class name or by using reference variable. |

Seven.java

|  |
| --- |
| import java.lang.\*; //import statement  class Dog  { static int s;  int a,b;  //instance method  void display() //e2  { System.out.println("Object state..........");  System.out.println("a="+a); //System.out.println("a="+e2.a);  System.out.println("b="+b); //System.out.println("b="+e2.b);  System.out.println("s="+s);  }  }  class Seven  {  public static void main(String args[])  {    //e1=1002  Dog e1=new Dog();// object creation  Dog.s=1000;  //e1 object initialization  e1.a=100;  e1.b=200;  e1.display(); //instance method calling  //e2=2002  Dog e2=new Dog();// object creation  //e2 object initialization  e2.a=300;  e2.b=400;  e2.display();  //e3=3002  Dog e3=new Dog();// object creation  //e3 object initialization  e3.a=500;  e3.b=600;  e3.display();  }  }  **Output:**  C:\Users\Madhu.K\Documents\JavaPrograms-1>java Seven  Object state..........  a=100  b=200  s=1000  Object state..........  a=300  b=400  s=1000  Object state..........  a=500  b=600  s=1000 |

What is private?

|  |
| --- |
| 1. Private is a keyword and also called as access modifier in Java but in C++ it is called as access specifier 2. If we define a field or a method as private that is not accessible outside the class i.e we can hide those members. |

Example on initializing objects using set methods

|  |
| --- |
| import java.lang.\*; //import statement  class Emp  {  static String company="TCS";  private int eno;  private String ename;  private float salary;  //when ever you write a function/method it is better to follow  //a naming convention: first word must be small and after that every word first letter must be capital.  void setEno(int en)//e1  { //en=101  eno=en; //e1.eno=en;  }  void setEname(String name)//e1  { //name="nikitha"  ename=name; //e1.ename=name  }  void setSalary(float sal)//e1  { //sal=100000.00  salary=sal; //e1.salary=sal;  }  void display()//e1  { System.out.println("Company is "+company);//Emp.company  System.out.println("Eno:\t"+eno); //e1.eno  System.out.println("Ename:\t"+ename);//e1.ename  System.out.println("Salary:\t"+salary);//e1.salary  }  }  class Eight  {  public static void main(String args[])  {  //e1=1002  Emp e1=new Emp();  Emp e2=new Emp();  Emp e3=new Emp();  //e1 object initialization  e1.setEno(101);  e1.setEname("Nikitha");  e1.setSalary(100000.00f);  //e2 object initialization  e2.setEno(102);  e2.setEname("Vinutna");  e2.setSalary(200000.00f);  //e1 object initialization  e3.setEno(103);  e3.setEname("Pranutna");  e3.setSalary(300000.00f);  e1.display();  e2.display();  e3.display();  }  }  **Output:**  C:\Users\Madhu.K\Documents\JavaPrograms-1>javac Eight.java  C:\Users\Madhu.K\Documents\JavaPrograms-1>java Eight  Company is TCS  Eno: 101  Ename: Nikitha  Salary: 100000.0  Company is TCS  Eno: 102  Ename: Vinutna  Salary: 200000.0  Company is TCS  Eno: 103  Ename: Pranutna  Salary: 300000.0 |