Commands:

* Javac – compile java program
* Java – run java program
* Javap – get methods of a class

Packages:

* Scanner – java.util.Scanner;
  1. nextInt()
  2. nextFloat()
  3. nextByte()
  4. nextLine()
  5. nextDouble()
  6. hasNextInt()
  7. useRadix(num)
* String – java.lang.String;
  1. length()
  2. toLowerCase()
  3. toUpperCase()
  4. trim()
  5. substring(int begin,int end)
  6. replace(char old, char new)
  7. startsWith(String s)
  8. endsWith(String s)
  9. equals(String s)
  10. equalsIgnoreCase(String s)
  11. compareTo(String s)
  12. valueOf(int i)

Notes:

* Datatypes
  1. Primitive Datatypes
     + Integral
       - Byte - 1
       - Short - 2
       - Int - 4
       - Long - 8
     + Floating point
       - Float - 4
       - Double - 8
     + Char - 2
     + Boolean – Min 1 bit or Max 1 byte
* Variables – Naming Rules
  1. Case sensitive
  2. Contains alphabets, numbers , \_, $
  3. Starts with Alphabet, \_, $
  4. Should not be a keyword
  5. Should not be a class name, If class is also in use
  6. No limit on length of name
  7. Follow Camel Cases
* Literals
  1. 5 is an Integer literal
  2. 153.75 is double literal
  3. “Java” String literal
  4. ‘java’ character literal
  5. 5L long literal
  6. 2.5f float literal
  7. True/false Boolean literal
  8. To print a decimal number in binary or octal or hexadecimal form use method Integer.toBinaryString(x);
  9. Floating point numbers are represented as 16352E-2 which is 163.52 (16352 x 10-2)
  10. Characters are Represented as ASCII (American standard code for information interchange) codes
  11. Every language in the world has a code called as UNICODE
  12. ACSII is a subset of UNICODE
  13. Java uses UNICODE to support multiple languages that is the reason character size in java is 2 bytes
* Interpreter and Compiler- language can either be compiled or interpreted or combination of both called as hybrid language.
  1. Interpreter translates the code line by line
     + JavaScript is an interpreted language
  2. Compiler translates all at once
     + C/C++ are compiled language
  3. Java is a hybrid language that is compiled and interpreted







Execution Engine has JIT/Interpreter and Garbage collector

* Operators
  1. Increment/Decrement Operators
     + ++,--
  2. Arithmetic Operators
     + +, - , %, \*, /
  3. Bitwise Operators
     + &,|,~,^,<<,>>,>>>
  4. Relational Operators
     + <,>,<=,>=,!=,==
  5. Logical Operators
     + &&,||,!
* Coercion and Casting
  1. Type coercion is the automatic or implicit conversion of values from one data type to another Operators
  2. Type casting explicit conversion of values from one data type to another Operators
* Bitwise
  1. X=0b1000
  2. X<<2 = 32
  3. X>>2 = 2
  4. X>>1 = 4
  5. X>>>1 = 21471231
* Bit Masking and Bit Merging
* System.out.printf(“Hello”); -- same as c/c++ printf();
* System.out.printf(“%d”, a);
  1. %+d – shows the sign
  2. % (d – represents in brackets
  3. %1$d – argument index
  4. %6f – width
  5. %6.2f – number of places before and after decimal
  6. %20s – width
  7. %s20 – spaces are left aligned
* Regular Expressions







* Arrays:
  + Datatype [] arrayname = new Datatype[size]; -- one dimensional array
  + For(int X: in A){} – for each loop
  + Array initialization Datatype [] arrayname={};
  + Jagged array
    - Multidimensional array with each array of different size
    - Int A[][]
    - A=new int[3][];
    - A[0] = new int[2];
    - A[1]=new int[4];
    - A[2]=new int[3];
* Methods
* OOP
  + Principles of OOP
    - Abstraction
    - Encapsulation
    - Inheritance
      * Deriving a new class from existing class is known as inheritance
      * Specialization in java is achieved using inheritance
    - Polymorphism
* **Generalization** In Generalisation group of classes are referred with Super class with single name. Generalisation means **Bottom Up.** In Generalisation A Super Class Is made by Grouping Multiple Sub Classes. Generalisation is achieved using **Interfaces.**
* **Specialisation** a new Sub Class is Generated by borrowing the features of existing concrete class and adding new features to it. Specialisation means **Top Down.** In specialisation a new Class is derived from an existing Super Class. Specialisation is achieved using **Inheritance.**
* Dynamic method dispatch
  + Used to achieve runtime polymorphism
  + Super sup = new Subclass ();
  + Creating a subclass object with superclass reference
  + Methods of subclass Is called as methods are called based on objects
  + Only the methods in super class overloaded in subclass can be used
* Abstract Classes
  + An abstract class can be a super class but much of the class functionality maybe undefined or unimplemented. This gap has to be filled by the programmer in the subclass.
  + Abstract methods start with abstract keyword and the class that has at least one abstract method is an abstract class
  + A class can be an abstract class even if it does not have at least one abstract method;
* Interface is a special case of an abstract class, which contains all abstract methods. An interface specify what a class must do but not how to do.