**Core Java**

**Java Basics**

* Java is strongly typed
* Java is a case sensitive language. Eg : num = 1, NUM = 1 are different

**Language fundamental** :

1. Identifiers
2. Reserved words
3. Data types
4. Literals
5. Array
6. Types of variables
7. Var arg
8. Main method
9. Command line arguments
10. Java coding standards

**1.IDENTIFIER**

* a name in java program is called identifer. Which can be used for identification purpose.
* It can be mehod name, variable name. Class name or lable name.

Rules for identifier

* Identifier are contains a to z, A to Z, 0 to 9, $, \_
* Identifier can’t start with digit not valid
* There is no length limit for java identifiers. But it is not recommended to take too lengthy identifiers.

**2. Reserved words**

* In java some words are reserved to represent meaning or functionality those are called reserved words.it is only lower case letters.
* We can’t use reserved words as identifiers eg: if, static etc
* Total 53 reserved words there
* 50 keywords
* 3 literals
* If reserved word associated with functionality it js keywords
* If reserved words only represents values are reserved literals - true, false, null, null is default value for object.
* In 50 keywords 48 are used and 2 un used key words.
* Unused keywords goto, const
* Used keywords are if, else, etc

Data type keywords

1. Byte
2. Short
3. Int
4. Long
5. Float
6. Double
7. Boolean
8. Char

Flow control keywords

1. If
2. Else
3. Switch
4. Case
5. Default
6. Do
7. While
8. For
9. Break
10. Continue
11. Return

**Keywords for modifiers**

1. Public
2. Private
3. Protected
4. Static
5. Final
6. Abstract
7. Synchronized
8. Native
9. Strictffp
10. Transient
11. Volatile

Exception handling keywords

1. Try
2. Catch
3. Finally
4. Throw
5. Throws
6. Assert

Class related Keywords

1. Class
2. Interface
3. Extends
4. Implements
5. Imports
6. Package

Object related keywords

1. new
2. instanceof
3. super
4. this

Void return type keyword

1. Void

Enum keyword

Used to define group of named constants.

**3.Data types**

* Every variable and expression in java has some data type
* Each and every data type is clearly defined, every assignment should be checked by compiler for type compatibility
* Because of the above reason we can conclude java language is strongly typed programming language.
* Java not supporting operator overloading and multiple inheritance etc
* It is pure oops, if it is pure oops means , everything should be object but we are depends on primitive data type.

**Primitive data types**

1. Numeric data type - to define numbers

* Integral Type - number without decimal point values

1. byte -
2. short -
3. int-
4. long-

* Floating point data type- to represent decimal point types

1. float-
2. double-
3. Non Numeric data type - data type not relevant to numbers

* char-
* boolean-
* If we assigned the value by out of range compiler throws **possible loss of precision error**
* If we assigned the value by different type compiler throws **Incompatible type**

**Byte :**

* Size is i bye and range is -128 to128.
* If we want to handle the data in files or trying to send the data across network we need to use byte data type , because both file and network are supported streams.
* If we want to handle the data in the form of stream either file or network we need to use byte.
* Two types streams there character stream, byte stream

**Short**:

* It is most rarely used data type in java
* Size is 2 bytes, range is -2 pow(15) to 2 pow (15) - 1 i.e -32768 to 32767
* At beggining of java 16 bit processor is very popular, i.e instructuion length is 16 bit , means data will be represent as 16 bit, so used short data type
* Short is best suitable for 16 bit like 8085 processor but now these processor out dataed so we are not using short

**Int:**

* Size : 4 byte range: -2 pow(31) to 2 pow (31)
* It is most commonly used data type in int.
* If assigned value is out of range compiler throws Integer number two large error
* If int x = 2147483648L compiler throws possible loss of precision found long required int
* If int x = true compiler throws incompatabile types error.

**Long**

* Sometimes int may not enought to hold big values then we should go for long types
* Eg: amount of distance traveled by light in 1000 days, to hold this value int may not enough we should go for long data type

**Float**

* It will use to refer 5 to 6 decimal places occuracy. It is less occuracy
* It follows single precision
* Size 4
* float f= 123.456f

**Double**

* It is used to refer 14 to 15 decimal places of occuracy
* It follows double precision
* Size 8
* double d = 123.456
* By default every floating point literal is double type

**Boolean**

* Size is not appliccable in java, virtual machine dependent
* Range also not applicable but allowed values are true or false
* boolean b=0 thrrows c.e as incompatable type
* boolean b = True throw CE as cannot find syymbol variable True
* That is it will take as a variable
* Default value is false

E.g :1

int x=0;

If(x){

S.O.P(x);

}

E.g:2

while(1){

S.O.P("hello")

}

Both are throws CE as found int required Boolean

**Char**

* Size 2 byte
* Java is unicode base language. Old language are like c, c±± ascii code base
* In old language allowed char are less than or equal to 256 for this 1 byte enough
* In java additional to these 256 char we can use worldwide alphabets
* So we used 16 bits
* Default value is 0 which represent space char.

Byte (1) -> short (2) ->

Int (4) -> long (8) -> float (4) -> double ( 8)

Char (2) ->

* Here long is 8, float is 4 bytes bcz internal representation for both are same
* Instance variables are depend on objects only, so we cant access instance variable from static directly

**4.Literals**

* It is a constant value which can be assigned to variable
* int x= 10, 10 is literal
* long x= 10, 10 is decimal literal
* int x= 010, 10 is octal literal
* int x= 0X10, 10 is hexadecimal literal

1. **var args**

* **it is also known as variable arguments**
* **if we don’t know the number of arguments of methods, we can use this var arg**
* **eg:**

**int add(int…numbers){**

**}**

**We can call this method by any arg like below,**

* **add(10) or add(10,20) or add(20,3,7)**
* **this var arg is similar to [] – single dimension array**
* **so we can call like below**

**eg: int[] a={10,20,30,40};**

**add(a);**

* **our var arg method will accept both array object and individual elements, if we declare the argument as [] it will accept only array obj and not elements.**

**Method Chaining:**

**1. in String, String buffer, String Builder most of the method returns same objects**

**2. after invoke any method we can invoke another method, which forms method chaning**

**3, ex: StringBuilder sb = new StringBuilder();**

**Sb.append(“durga”).appernd(“system”).reverse();**

**4.now method execution will take left to right**

**5.**

**Garbage Collection**

**1.Introduction**

* **The purpose of garbage collection is to delete unused objects**
* **Sun people provided one feature which always run in background (Daemon thread) and destroy useless objects.**
* **Bcz of this chance of failing java program with memory problems is very very low. This feature is Garbage Collector.**
* **Hence the main objective of GC is to destroy useless objects**

**2.make object to eligible for GC**

* **If an object does not have any reference variable then it is eligible for GC**
* **There are 4 ways to make object eligible for GC**

1. **Nullify the reference variable/assign null to reference variable**

**String s= new String();-- object created.**

**s=null; now s not pointing to any object so. The created object is eligible for GC.**