Java Identifiers, Data Types & Variables

1. Java Identifiers:

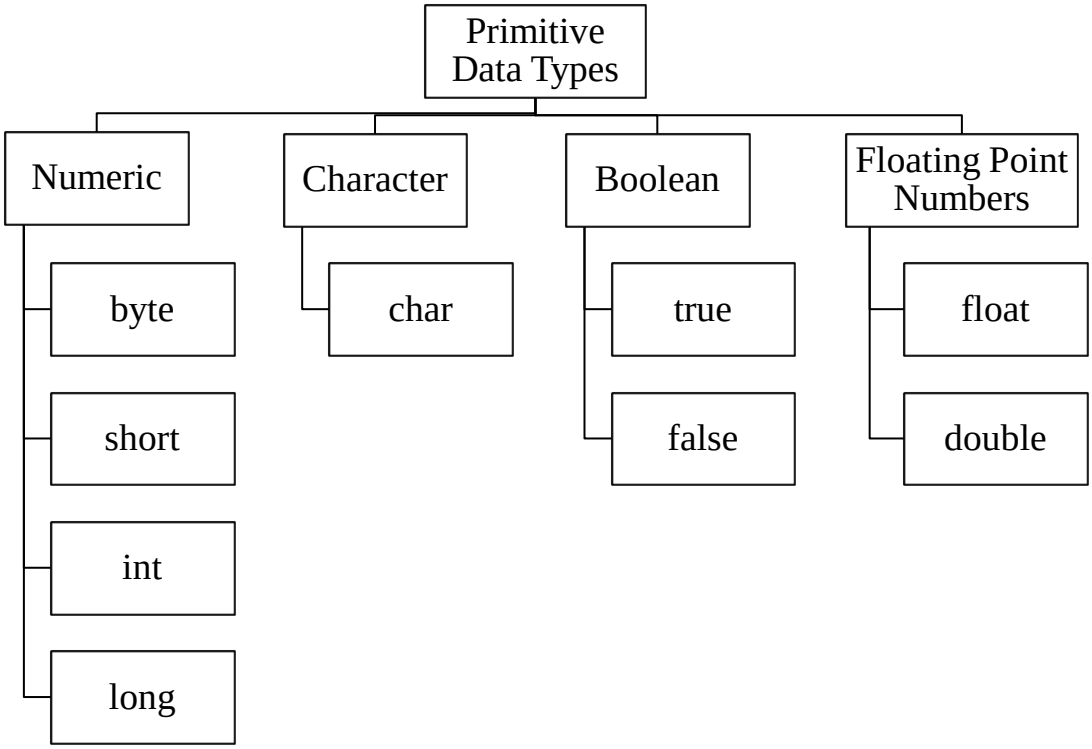
- Identifiers are name given to a class, variable or a method.

```
public class TestingShastra {  //TestingShastra is an identifier for class
      char ch = 'A';      //ch is an  identifier for variable

      public void display() {  //display is an identifier for method
            int x = 10;

      }
}
```

- **Rules to define Identifiers:**
  - Java identifiers are case sensitive.
  - Identifier name should not start with/contain special character except (_) underscore and ($) Dollar sign.
  - Variable name should not start with integer. Intermediate and trailing integers are allowed.
    Ex. int 123abc //*Not allowed*
    int abc123 //*Allowed.*
  - Variable name should not have space
    Ex. int abc xyz //*Not allowed*
  - If you have multiple words for a variable then first letter of a first word should be small and next subsequent word's letters should be capital (Ex. int abcXyz)
  - There is no limit for java identifiers. But it is recommended to use identifier length<15.
  - Java reserved words are not allowed to use as identifiers.

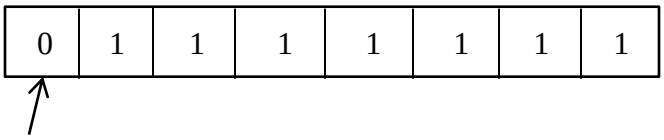
2. Java Data Types:

- Java strictly defines type of data it may process. Data types states different types of values to be stored in variable. Data types supported by java are basically divided in two types:
    - Primitive Data Types
    - Non-Primitive Data TypesbytrNumer
    - N



Numeric Data Types:

- **Byte:**
    - All integer data types are signed in java
    - Bytes is 8-bit long (1 byte)
    - Max size=127 (2^7-1)
    - Min size=-128 (-2^7)
    - Best to use where we want to read or write data to/from file.
    - Ex. byte b=125 *//Allowed*
      byte b=133 *//Not Allowed. Exceeds range. Loss of data may happen*



Sign bit (0- Positive1-Negative)

- **Short:**
    - Short is 2-byte long.
    - Max size = 32767 (2^{15}-1)
    - Min Size= -32768 (-2^{15})

- Ex. short sh=234 //*Allowed*
- Least used.

- **Int:**
  - Int is 4-bytes long
  - Most commonly used.
  - Max Size= 2,147,483,647 (2^{31}-1)
  - Min Size= -2,147,483,646 (-2^{31})
  - Most commonly used in loops and conditional statements.
  - Ex. int a=10,b=5 //*Allowed*
- **Long:**
  - Long is 8-byte long.
  - Max Size= 9,223,372,036,854,775,807 (2^{63}-1)
  - Min Size= -9,223,372,036,854,775,806 (-2^{63})
  - It is used where int is not sufficient to store data. Ex. Calculate number of seconds in 10 years. (10x365x24x60x60)

Floating Point Data Types:

Floating-point data types are used to represent *real* numbers. They are commonly used when we wish to save data with precision. Example- square root of some number

- **Float:**
  - Float is 4-bytes long.
  - float specifies a single-precision value that uses 32 bits of storage.
  - Max Size= 3.4e+038
  - Min Size= 1.4e–045
  - Ex. float pi=3.14f //*append f at the last*
- **Double:**
  - Double is 8-bytes long.
  - double specifies a double-precision value that uses 32 bits of storage.
  - Max Size= 1.8e+308
  - Min Size= 4.9e–324
  - It is useful when we want to maintain accuracy with large precision numbers.
  - Ex. double d=3.142707

Character Data Types:

- **Char:**
  - Char is 16-bit wide in Java
  - Java uses *Unicode* to represent characters.
  - It's range is 0 to 65,536
  - There are no negative characters
  - It can save all characters from worldwide lanuages.
  - Ex. char ch='A' //*Value should be declared in single quotes*

Boolean Data Types:

- **Boolean:**
  - It is used to save logical values.
  - It can save only **true** and **false** value.
  - It is mainly used in conditional statements like, if and if-else.
  - Ex. boolean b=true
    boolean b=false

3. Variables:

- Variables are memory locations having a name
- Variables are used to store data.
- Variables are combination of data type, identifier and optional initialization.
- All variables have scope and lifetime. Based on their scope they are divided into three types.
  i. Instance Variable
  ii. Static Variable
  iii. Local Variable

I.Instance Variable:

- A variable declared inside class is an instance variable
- For every object, separate copy of instance variable will be created.
- Their life time is same as objects life time.
- Instance variables are loaded in memory at the time of Object creation.
- Instance variables are declared in class and outside of method, block and constructors.

- Instance variables are accessed using class's object.
- Ex.

```
1.    public class Test {
2.       static int z=30;
3.       int x=10;  //Instance Variable
4.       public void m1(){
5.           int y=20;  //Local variable
6.       }
7.       Public static void main(String[] args){
8.           Test t1=new Test();
9.           Test t2=new Test();
10.          System.out.println(x);
11.          System.out.println(t1.x);
12.          System.out.println(Test.z);
13.          System.out.println(z);
14.      }
15.   }
```

- In above example, 'x' is an instance variable.
- It cannot be accessed from static method. That's why line number 10 will give Compile Time Exception.
- It can be accessed using object of a class. Hence line number will work fine.
- Object t1 and t2 will have separate copies of x in memory.

II. Static Variable:

- When we don't want to create separate copies of variables for different objects, then we can use static variables.
- Static variables have single copy in memory. They are single copy storage.
- Static variables are independent of object.
- We can declare static variable using 'static' modifier.
- Static variables are loaded into memory at the time of class loading. And destroys when class destroys.
- Static variables are defined in class and outside of method, block or constructors.
- Static variables can be accessed using class name or by object reference. In above program observe line no. 12.
- To access static variable in same class, it is not required to access using class name of reference variable. We can directly call static variables in same class. Observe line no. 13.

- If we change value of instance variable for one object, it will not have any effect on another object's copy of the instance variable. Because for every object there is a separate copy of instance variable.
- But in case of static variable, if we change its value it will be reflected to each and every object of the class as it is single copy storage.

III.Local Variable:

- Variables declared inside methods, blocks or constructors are called as local variables.
- Local variable's scope is limited to the method, block or constructor inside which it is defined.
- Local variables are stored inside stack because they are temporary.
- Local variables are loaded inside memory when the corresponding method is loaded. And destroys when the method is completed its execution.
- In program on previous page, y is a local variable.
- For local variables, JVM won't provide any default value. Compulsorily we should initialize it. But for instance and static variables JVM will provide default values.
- The only allowed modifiers for local variable are '*default*' and '*final*'. Public, private, protected, static are not allowed.
- We cannot access local variables using object reference.

Quiz

1. Which of the following is valid identifier declaration?
    a. Testing123Shastra
    b. 123TestingShastra
    c. Testing@Shastra
    d. Testing-Shastra
    e. Testing$Shastra
    f. Integer
    g. char
2. Which of the following is valid declaration for byte?
    a. byte b=100
    b. byte b=135
    c. byte b=12.34
    d. byte b=true
3. What are the sizes of byte, short, int and long?
4. Which of the following declaration for boolean is correct?
    a. boolean b=0
    b. boolean b=true
    c. boolean b=True
    d. boolean b='false'
    e. Boolean True=true
5. Which of the following statements are valid?
    a. Local variables can be static.
    b. Static variables have separate copies for every object.
    c. Instance variables are class level variables
    d. Static variables can be accessed using Object reference only.