Java

Module 2

Java Language

Author
Srinivas Dande

My JAVA LEARNING CENTER

2.1 Character Sets

Alphabets	A - Z / a - z
Digits	0 - 9
Special Characters	+ - * & % $! < > { } = ~ ^ etc

2.2 Keywords

- Keywords are simple English words which are having predefined meaning in Java Programming Language.
- Meaning of these keywords can't be modified.
- Keywords are also called as Reserved Words.
- All the keywords are defined in Lower Case.
- We can't use keywords as names for variables, methods, classes, or as any other identifiers.
- true, false, null are not keywords but reserved as Literals.
- There are 49 keywords defined up to Java 2.
- enum is the new keyword added from Java 5 so there are 50 keywords from Java 5.
- const & goto are the keywords but no implementation available. You can't use these keywords in Java Program.

List Of Keywords			
Data types (8)	boolean short float	byte int double	char long
Class & Object (9)	class extends super	interface implements new	enum this instanceof
Access Modifier (3)	private	protected	public
Modifier (9)	final synchronized static	native transient const *	abstract volatile strictfp
Package (2)	package	import	
Control Statements (12)	if case while continue	else default break return	switch do for goto *
Exception Handling (6)	try throw	catch throws	finally assert
Other Data type (1)	void		

2.3 Identifiers

- Identifiers are the names those will be used to identify the programming elements like classes, methods, variables, etc uniquely.
- Identifiers are also known as User Defined Words.

Rules to follow when you define an Identifier:

- Identifier can contain Alphabets, Digits, and two special symbol i.e. Dollar ($), Underscore (_).
- First character of an identifier must be an Alphabet or Dollar ($) or Underscore (_).
- Keywords or Reserved words can't be used as Identifier.

Valid Identifiers	Invalid Identifiers
Hello Int getClassName studentEmail TOTAL_FFE	1stClass true student number student-email int

2.4 Data types

- Data type represents:
  - Type of data you want to use
  - Amount of memory allocation required for your data.

There are two types of data types

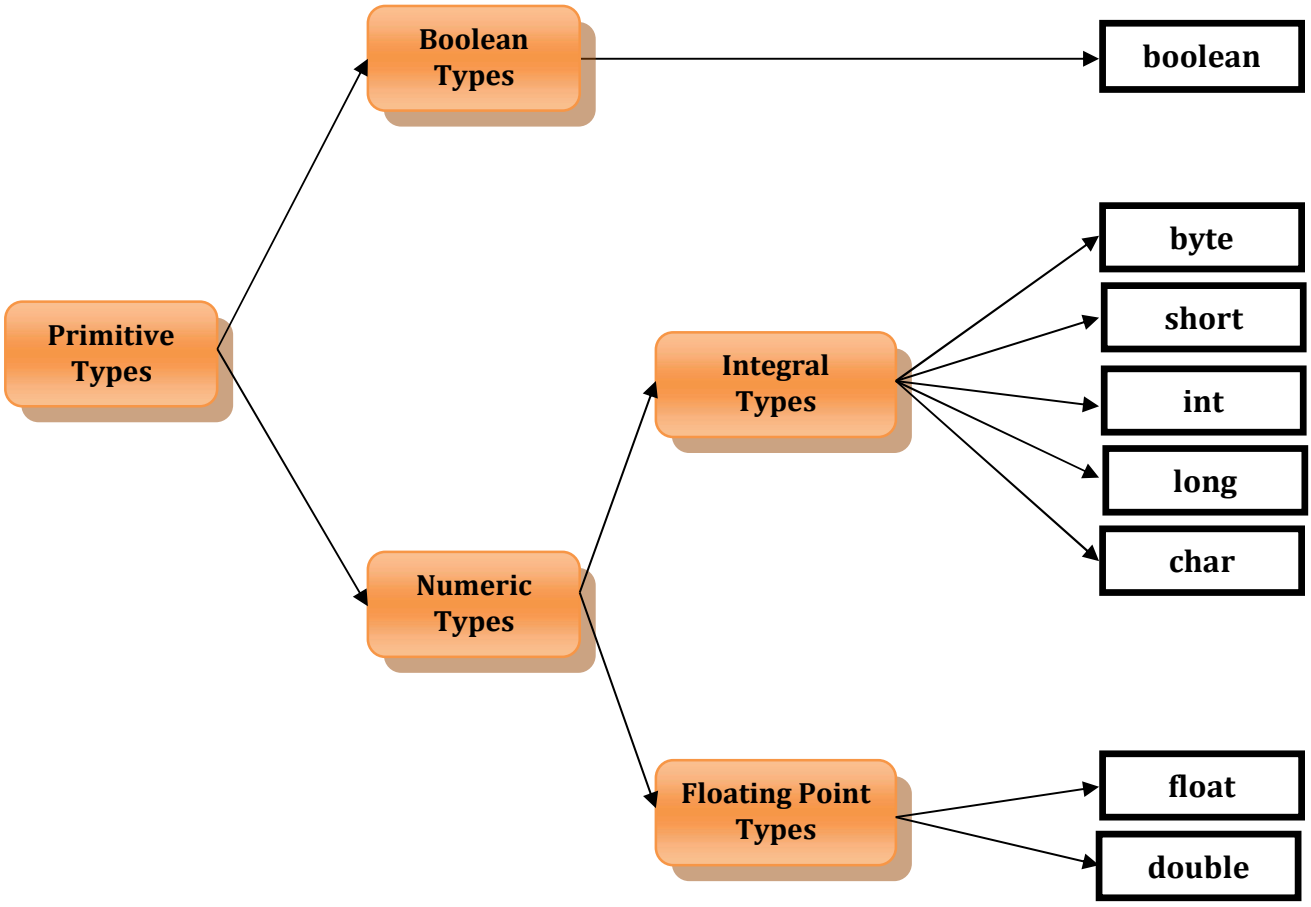
1) Primitive Data Types
2) User Defined Data Types

2.4.1 Primitive Data types

- There are 8 primitive types available in Java.
- 8 keywords are defined to represent 8 primitive data types.

- **Following are primitive data types:**

boolean	byte	char	short
int	long	float	double



2.4.2 User Defined Data types

- There are four types of User Defined Data types:
  - Class type
  - Interface type
  - Enum type (From JAVA 5)
  - Annotation type (From JAVA 5)

Type	Size		Default Value	Min Value	Max Value
	Byte	Bits			
boolean	N/D	N/D	false	true or false	
byte	1	8	0	-2^7 (-128)	2^7 -1 (127)
char	2	16	ASCII – 0 Unicode - \u0000	0	2^{16}-1 (65535)
short	2	16	0	-2^{15} (-32,768)	2^{15}-1 (32,767)
int	4	32	0	-2^{31} (-2147483648)	2^{31}-1 (2147483647)
long	8	64	0	-2^{63}	2^{63}-1
float	4	32	0.0	1.4E-45	3.40E38
double	8	64	0.0	4.9E-324	1.79E308
Any Reference Type	8	64	null	Reference of the corresponding type object	

2.5 Variables

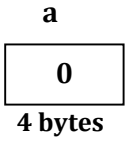
- Variable is the container which holds user data.
- Memory will be allocated for the variable while executing the program.
- Value of the variable can be changed any number of times during the program execution.

Syntax:

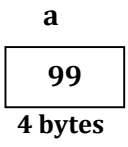
```
<Data type> <varName>;
<Data type> <varName> = <value>;
```

Ex:

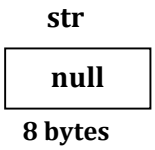
```
int a ;
```



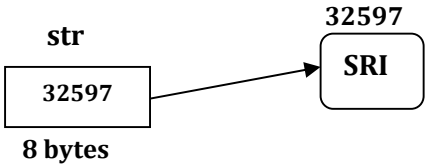

```
int a = 99;
```




```
String str;
```




```
String str="SRI";
```



Types of Variables

There are two types of variables based on data type used to declare the variable.

1) Primitive Variables
2) Reference Variables

2.5.1 Primitive Variables

- Variables declared with primitive data types are called as primitive variables.

Ex:

```
int a;
int b = 99;
double d1;
double d2=9.9;
```

2.5.2 Reference Variables

- Variables declared with user defined data types are called as reference variables.

Ex:

```
String str1;
String str2 = "JLC";
```

Lab1.java

```
class Hello{
boolean b1;
byte b2;
short s;
int a;
long b;
float f;
double d;
String str;
Hello h;
void show(){
System.out.println(b1);
System.out.println(b2);
System.out.println(s);
System.out.println(a);
System.out.println(b);
System.out.println(f);
System.out.println(d);
System.out.println(str);
System.out.println(h);
}
}
class Lab1 {
public static void main(String[] as) {
Hello h = new Hello();
h.show();
}
}
```

Lab2.java

```
class Hello{
char ch;
void show(){
System.out.println(ch == 0);
System.out.println(ch == ' ');
System.out.println(ch == '\u0000');
}
}
class Lab2 {
public static void main(String[] args) {
Hello h=new Hello();
h.show();
}
}
```

Primitive Variables	Reference Variables
1. Variables declared with primitive data types are called as primitive variables.	1. Variables declared with reference data types are called as reference variables.
2. Memory allocation for primitive variables depends on the primitive data types.	2. Always 8 bytes of memory will be allocated for reference variables.
3. Default value for primitive variables depends on primitive data types.	3. Always null will assigned as default value for reference variables.
4. Primitive variables hold valid literals or value in the allocated memory.	4. Reference variables hold either null or address of an object in the allocated memory.

Types of Variables

There are three types of variables based on the scope of the variables.

1) Instance Variables
2) Static Variables
3) Local Variables

1) Instance Variables :

- Variables declared in the class without using static keyword are called as Instance variables.

2) Static Variables

- Variables declared in the class using static keyword are called as Static variables.

3) Local Variables

- Variables declared in the member of the class like method etc are called as Local variables.

```
class Hello{
        int a;          ---> Instance Variable
        static int b;   ---> Static Variable
        void show(){
                int c;  ---> Local Variable
        }
}
```

There are various syntaxes for declaring variables

1) Declaring a single variable in one variable declaration statement.

Syntax:

[modifiers] <Data type> <varName>;

Ex:

```
byte b;
int a;
double d;
char ch;
String str;
Hello h;
etc
```

Lab3.java

```
class Lab3{
public static void main(String[] as) {
int a;
System.out.println(a);
}
}
```

Lab4.java

```
class Lab4 {
static int a;
public static void main(String[] as) {
System.out.println(a);
}
}
```

Lab5.java

```
class Lab5 {
int a;
public static void main(String[] as) {
System.out.println(a);
}
}
```

2) Initializing a single variable

Syntax:

<varName>= <value/expression>;

Ex:

```
b = 24;
ab = 98787;
d = 1234.456;
ch ='A';
str="JLC";
h=new Hello();
```

Lab6.java

```
class Lab6 {
public static void main(String[] as) {
String str;
int a;
str="JLC";
a=99;
System.out.println(str);
System.out.println(a);
}
}
```

3) Declaring and initializing a single variable in one variable declaration statement.

Syntax:
[modifiers] <Data type> <varName> = <value/expression>;

Ex:

```
int a = 99;
String str = "Srinivas";
double d = 999.99;
boolean b = true;
```

Lab7.java

```
class Lab7 {
public static void main(String as[]) {
int a = 99;
boolean b = true;
double d = 999.99;
String str = "Srinivas";
System.out.println(a);
System.out.println(b);
System.out.println(d);
System.out.println(str);
}
}
```

4) Declaring multiple variables of single type in one variable declaration statement.

Syntax:
[modifiers] <Data type> <var1>, <var2>, <var3>...;

Ex:

```
byte b1, b2, b3;
int a, b, c;
char ch1,ch2,ch3;
String str1, str2, str3;
```

5) Initializing multiple variables with same value

Syntax:
<var1>=<var2>=<var3>=<value/expression>;

Ex:

```
a = b = c = 9899;
str1 = str2 = str3 ="JLC"
```

Lab8.java

```
class Lab8{
public static void main(String[] as) {
int a, byte b;
System.out.println(a);
System.out.println(b);
}
}
```

Lab9.java

```
class Lab9{
public static void main(String[] as) {
int a, b, c;
a=b=c=99;
System.out.println(a);
System.out.println(b);
System.out.println(c);
String str1, str2, str3;
str1 = str2 = str3 ="JLC";
System.out.println(str1);
System.out.println(str2);
System.out.println(str3);
}
}
```

6) Declaring and initializing multiple variables of single type in one variable declaration statement.

Syntax:

[modifiers] <Data type> <var1>=<val1>,<var2>=<val2>,<var3>=<val3> ...;

Example:

```
int a=120,b=76,c=56;
String str1= "JLC", str2 = "SRI";
boolean b1=true,b2=false;
```

Lab10.java

```
class Lab10{
public static void main(String as[]) {
int a=120,b=76,c=56;
System.out.println(a);
System.out.println(b);
System.out.println(c);
String  str1= "JLC", str2 = "SRI";
System.out.println(str1);
System.out.println(str2);
}
}
```

2.6 Constants

- Constant is a special variable whose value can't be modified during the program execution.
- Constant is also called as final variable.

Syntax:

[modifier] final <data Type> <varName>=<value>;

Ex:

final int A=99;
final String STR="JLC";
final double D1=999.99;

- A variable that is declared as final and not initialized is called as blank final variable.

Syntax:

[modifier] final <dataType> <varName>;
<varName> = <value>;

Ex:

final int A;
A = 98;

- If the variable is declared as final and initialized in the same statement then in the class file compiler will replace that variable with actual value.

Lab11.java

```
class Lab11{
public static void main(String as[]) {
int a = 99;
System.out.println(a);
a = 88;
System.out.println(a);
}
}
```

Lab12.java

```
class Lab12{
public static void main(String as[]) {
final int A = 99;
System.out.println(A);
A=88;
System.out.println(A);
}
}
```

Lab13.java

```
class Lab13{
public static void main(String as[]) {
final int A;
System.out.println(A);
}
}
```

Lab14.java

```
class Lab14{
public static void main(String as[]) {
final int A;
A=99;
System.out.println(A);
}
}
```

Lab15.java

```
class Lab15{
public static void main(String as[]) {
final int A;
A=99;
System.out.println(A);
A=98;
System.out.println(A);
} }
```

Lab16.java

```
class Lab16{
public static void main(String as[]) {
const int a=99;
System.out.println(a);
}
}
```

Lab17.java

```
class Lab17{
public static void main(String as[]) {
final int A = 9;
System.out.println(A);
System.out.println(A + 90);
}
}
```

SUMMARY

Variables & Constants

1) Default value of char is either ASCII - 0 or UNICODE \u0000.
2) Default value of char is not space.
3) JVM will not provide default value for Local Variable.
4) Local variable must be initialized before using .
5) JVM will provide default value for static Variable.
6) Static variable can be accessed from Static method (main method).
7) JVM will provide default value for instance Variable.
8) Instance variable cannot be accessed directly from Static method (main method).
9) You can't declare multiple variables of different data types in single variable declaration statement.
10) You can declare multiple variables of same data types in a single statement.
11) You can assign same value to multiple variables in a single statement.
12) You can declare and initialize multiple variables of same data type in a single statement.
13) We can't declare two variables with same name in same scope.
14) Value of the variable can be changed any number of times during program execution.
15) Value of the final variable can not be changed.
16) We can't use const keyword to declare constant.

Assignment #1

Q1) How many keywords are available prior to Java 5?

Q2) What is the keyword added newly in Java 5?

Q3) Which of the following are valid Java keywords?

A) instanceof	**B) goto**	**C) null**	**D) assert**	**E) struct**
F) true	**G) Long**	**H) false**	**I) virtual**	**J) signed**

Q4) What are the keywords available which can't be used in Java Program?

Q5) What is an identifier? What are the rules to follow to define an identifier?

Q6) What is data type? How many types of data types available?

Q7) How many primitive data types available?

Q8) What is the Integral data type that will not allow negative value?

Q9) What is the default value of char data type?

Q10) Why the range of short and char are different even both required 2 bytes of memory?

Q11) How many types of User defined data types available up to JDK1.4?

Q12) How many types of User defined data types available from Java 5?

Q13) Is String a data type?

Q14) What is the default value of reference data type?

Q15) What are the possible values can be used for boolean type?

Q16) What is a variable?

Q17) How many types of variables are available as per the data type?

Q18) How many types of variables are available as per the scope?

Q19) What are the differences between Primitive variables and Reference variables?

Q20) What is the default value of local variable?

Q21) How can I declare multiple types of variables in one statement?

Q22) How can I declare multiple variable of same type in one statement?

Q23) How can I declare and initialize a variable in one statement?

Q24) How can I declare and initialize multiple variables of same type in one statement?

Q25) Can I change the value of variable during program execution?

Q26) When will the memory be allocated for the variables?

- **A) While compiling the code**
- **B) While executing the code**

Q27) Will JVM provide default value for static variable?

Q28) What is a constant?

Q29) What is the blank final variable in Java?

Q30) What will happen when you try to modify the value of the final variable?

Q31) Can I use const keyword to define the constant? If so what will happen?

Practice Test #1

Sr. No.	Question	Options	Answer

1	class Test1{
	public static void main(String as[]){
	int enum=9;
	System.out.println(enum);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 9
		D) Display 57	

2	class Test2{
	public static void main(String as[]){
	char charValue='A';
	System.out.println(charValue);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display A
		D) Display 65	

3	class Test3{
	public static void main(String as[]){
	byte b=128;
	System.out.println(b);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 128
		D) Display -128	

4	class Test4{
	public static void main(String as[]){
	int a=b=c=999;
	System.out.println(c);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 0
		D) Display 999	

5	class Test5{
	public static void main(String as[]){
	boolean valid=TRUE;
	System.out.println(valid);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display TRUE
		D) Display true	

6	class Test6{
	public static void main(String as[]){
	int _123=99;
	System.out.println(_123);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 99
		D) Display 123
		E) Display _123	

7	class Test7{
	public static void main(String as[]){
	String ins="null";
	System.out.println("String");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display null
		D) Display String	

8	class Test8{
	public static void main(String as[]){
	int $20=1200;
	System.out.println($20);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display $20
		D) Display 1200	

9	class Test9{
	public static void main(String as[]){
	byte $12_34=99;
	System.out.println($12_34);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 99
		D) Display $12_34
		E) Display 12.34
		F) Display 12_34	

10	class Test10{
	public static void main(String as[]){
	char CHAR='A';
	System.out.println(CHAR);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display A
		D) Display CHAR
		E) None of above	

11	class Test11{
	public static void main(String as[]){
	String String="JLC";
	System.out.println(String);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display JLC
		D) Display String	

12	class Test12{
	public static void main(String as[]){
	char ch;
	System.out.println(ch == 0);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display true
		D) Display false
		E) None of above	

13	class Test13{
	public static void main(String as[]){
	int a=10, b =a;
	System.out.println(a);
	System.out.println(b);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display
		10 10
		D) Display 10 0	

14	class Test14{
	public static void main(String as[]){
	int a= a =10;
	System.out.println(a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 10
		D) Display 2	

15	class Test15{
	public static void main(String as[]){
	System.out.println($20);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display $20
		D) Display 20
		E) Display 1200	

16	class Test16{
	public static void main(String as[]){
	int _123,_234;
	int val=_123=_234=90;
	System.out.println(val);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 123
		D) Display 234
		E) Display 90	

17	class Test17{
	public static void main(String as[]){
	int a= b, b = 10;
	System.out.println(a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 10
		D) Display 10 10
		E) Nont of above	

18	class Test18{
	public static void main(String as[]){
	final int ab;
	ab = 12;
	System.out.println(ab+10);
	System.out.println(ab);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 10 10
		D) Display 22 12
		E) Display 22 22	

19	class Test19{
	static String name;
	public static void main(String as[]){
	System.out.println(name);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display null
		D) Display 0
		E) None of above	

20	class Test20{
	String email;
	public static void main(String as[]){
	System.out.println(email);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display null
		D) Display 0
		E) None of above	

21	class Test21{
	public static void main(String as[]){
	String email;
	System.out.println(email);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display null
		D) Display 0
		E) None of above	

22	class Test22{
	public static void main(String as[]){
	int a=99;
	int b=a;
	System.out.println(a);
	System.out.println(b);
	b=88;
	System.out.println(a);
	System.out.println(b);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display
		99 99
		88 88
		D) Display
		99 99
		99 88
		E) None of above	

23	class Test23{
	public static void main(String jlc[]){
	final int AB=99;
	System.out.println(ab);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 99
		D) Display 0
		E) None of above	

24	class Test24{
	static char val;
	public static void main(String jlc[]){
	System.out.println(val == 0);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display true
		D) Display false
		E) Display 0	

25	class Test25{
	public static void main(String jlc[]){
	final int A;
	System.out.println('A');
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display A
		D) Display 0
		E) None of above	

26	class Test26{
	public static void main(String jlc[]){
	static int a;
	System.out.println(a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Display 0
		D) Display null	

27	class Test27{
	public static void main(String jlc[]) {
	String str1, str2;
	String str3 = str1 = str2 = "JLC";
	System.out.println(str1);
	System.out.println(str2);
	System.out.println(str3);
	} }
		A) Compilation Error
		B) Runtime Error
		C) null null null
		D) JLC JLC JLC
		E) null null JLC	

28	class Test28{
	public static void main(String as[]) {
	int b, c;
	int a = b = c = 88;
	System.out.println(a);
	System.out.println(b);
	System.out.println(c);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 0 0 88
		D) 88 88 88	

29	class Test29{
	public static void main(String jlc[]) {
	int a, b;
	int c = 10, a = 20, b = 88;
	System.out.println(a);
	System.out.println(b);
	System.out.println(c);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 0 0 88
		D) 88 88 88	

30	class Test30{
	public static void main(String jlc) {
	byte b=128;
	System.out.println(b);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 10
		D) No Output	