Why we need data types in Java?

The Data Types in Java is basically used to store the data temporarily in the computer through a program. In the real world, we have different types of data like integer, floating-point, character, string, etc. To store all these different types of data in a program to perform business-related operations, we need the data types.

What is a data type in Java?

The Data types are something which gives information about

1. **Size** of the memory location.
2. The **range of data** that can be stored inside that memory location
3. Possible **legal operations** that can be performed on that memory location.
4. What **types of results** come out from an expression when these types are used inside that expression.

The keyword which gives all the above information is called the **data type**

Classification of Java Data Types:

The **Data Types in Java** specifies the size and type of values that can be stored in an identifier or variable. The **Java** language is rich in its data types. Data types in Java are classified into two types:

1. **Primitive Types:** Examples: Integer, Character, Boolean(true , or false), and Floating Point.
2. **Non-primitive Types:** Examples: Classes, Interfaces, and Arrays.

Java is Object Oriented. However it is not considered as pure object oriented as it provides support for primitive data types (like int, char, etc)

Java Primitive Data Types
Those data type whose value and size has fixed.
There are 8 types of primitive data types:

boolean data type

byte data type

char data type

short data type

int data type

long data type

float data type

double data type

Data Type	Default Value	Default size
Boolean	false	1 bit
Char	'\u0000'	2 byte
Byte	0	1 byte
Short	0	2 byte
Int	0	4 byte
Long	0L	8 byte
Float	0.0f	4 byte
Double	0.0d	8 byte

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs**. The keywords const and goto are reserved, even though they are not currently used**. **true, false, and null might seem like keywords, but they are actually literals**; you cannot use them as identifiers in your

programs.

abstract	continue	for	new	switch
assert* **	default	goto*	package	synchronized
boolean	do	if	private	this
Break	double	implements	protected	throw
Byte	else	import	public	throws
Case	enum****	instanceof	return	transient
Catch	extends	int	short	try
Char	final	interface	static	void
Class	finally	long	strictfp**	volatile
const*	float	native	super	while

*	**not used**
**	added in 1.2
***	added in 1.4
****	added in 5.0

program Structure

```
class   <ClassName>
 {

      public static void main(String args[] )
       {



       }


  }
```

```
public static void main(String[] args)
  {
     System.out.println("");
```

```
}
```

the point from where the program starts its execution or simply the entry point of Java programs is the **main()** method

Public: It is an *Access modifier*, which specifies from where and who can access the method. Making the *main()* method public makes it globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class

Static: It is a *keyword* which is when associated with a method**, makes it a class related method**. The *main()* method is static so that **JVM can invoke it without instantiating the class**. This also saves the unnecessary wastage of memory which would have been used by the object declared only for calling the *main()* method by the JVM

Void: It is a keyword and used to specify that a method doesn't return anything. As *main()* method doesn't return anything, its return type is *void*. As soon as the *main()* method terminates, the java program terminates too. Hence, it doesn't make any sense to return from *main()* method as JVM can't do anything with the return value of it.

main: It is the name of Java main method. It is the identifier that the JVM looks for as the starting point of the java program. **It's not a keyword**

String[] args: It stores Java *command line arguments* and is an array of type *java.lang.String* class. Here, the name of the String array is *args* but it is not fixed and user can use any name in place of it.

Output Statement:

System.out.println(“Abhay kUmar”) // generate output always in new line

System.out.print (“Abhay kUmar”) // generate output always in Same line

System.out.printf(“Abhay kUmar”) // generate output always in formatting using c library)