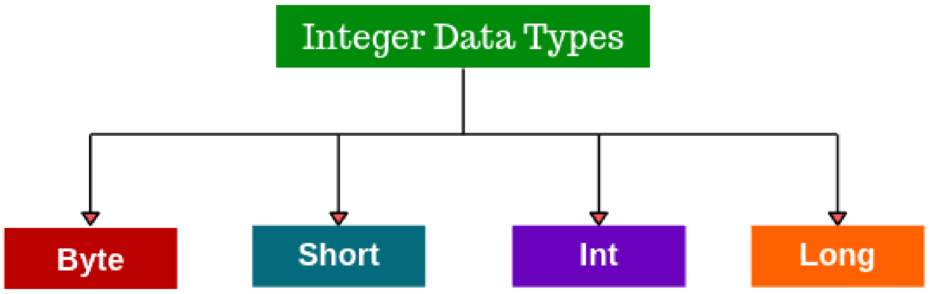
Integer type data:

Integer types of data represent integer number without any fractional parts or decimal points.

For example: age of a person, number of students in a class, distance between planets, distance between galaxy, etc.,

In Java, to store Integer type data we have four data types.



Now the question arises that to store Integer type data why do we require four data types??

Before that let's have a look on a formula

It is important for one to understand that, given a certain amount of memory how much data can be stored inside it. There is a certain maximum value and minimum value that can be stored and the formula to find it is:

*For **n bits**: The minimum value that can be stored is* -2^{n-1}

The maximum value that can be stored is $+2^{n-1} - 1$

1) **byte** is the smallest integer data type which has the least memory size allocated.
Assume, if we create a byte type variable as **byte a;**
It means in the memory **one byte** is allocated and it is referred as **a**.

To convert byte to bits we have, **1 byte = 8 bits.**
And therefore, the minimum value is -2^{8-1} = -2^{7} = -128
The maximum value is $+2^{8-1} - 1$ = $+2^{7}$-1 = +127

Now let's see few examples to understand the above topic.

Example-1:

```
class Demo
{
	public static void main(String[] args)
	{
		byte a;
		a = 127;
		System.out.println(a);
	}
}
```

Output:

```
 127
```

Example-2:

```
class Demo
{
	public static void main(String[] args)
	{
		byte a;
		a = 128;
		System.out.println(a);
	}
}
```

Output:

Compilation error (because this is out of range).

Example-3:

```
class Demo
{
	public static void main(String[] args)
	{
		byte a;
		a = -128;
		System.out.println(a);
	}
}
```

Output:

```
 -128
```

Example-4:

```
class Demo
{
	public static void main(String[] args)
	{
		byte a;
		a = -129;
		System.out.println(a);
	}
}
```

Output:

Compilation error (because this is out of range).

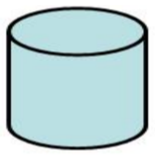
Which means, one value more than +127 or one value less than -128 cannot be stored but any value between the range of -128 to +127 can be stored using a byte type variable.

Therefore, the age of a person, the current month of the year etc., can be stored using byte data type.

2) short is a signed 16-bit byte.

Assume, if we create a short type variable as **short a;**

It means in the memory **two bytes** is allocated and it is referred as **a**.


short
2

Converting byte to bits: **2 bytes = 16 bits.**

And therefore, the minimum value is -2^{16-1} = -2^{15} = -32768

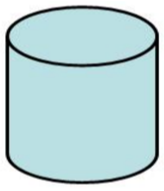
The maximum value is $+2^{16-1}-1$ = $+2^{15}$-1 = +32767

Which means, one value more than +32767 or one value less than -32768 cannot be stored but any value between the range of -32768 to +32767 can be stored using a short type variable.

Therefore, salary of a fresher, height of Mount Everest etc., can be stored using short data type.

3) int is a signed 32-bit byte. It is the most commonly used integer type. In addition to other uses, variables of type int are commonly employed to control loops and to index arrays.

Assume, if we create an int type variable as **int a;**
It means in the memory **four bytes** is allocated and it is referred as **a**.

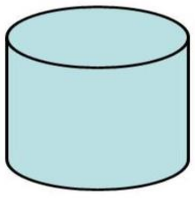


Converting byte to bits, **4 bytes = 32 bits.**
And therefore, the minimum value is -2^{32-1} = -2^{31} = **-2,147,483,648**
The maximum value is $+2^{32-1} - 1$ = $+2^{31}$-1 = **+2,147,483,647**

Which means, one value more than +2,147,483,647 or one value less than -2,147,483,648 cannot be stored but any value between the range of -2,147,483,648 to +2,147,483,647 can be stored using an int type variable.

Therefore, distance between planets etc., can be stored using int data type.

4) long is a signed 64-bit byte and is useful for those occasions where an int type is not large enough to hold the desired value. The range of long is quite large. This is useful, when big whole numbers are needed.
Assume, if we create an long type variable as **long a;**
It means in the memory **eight bytes** is allocated and it is referred as **a**.



Converting byte to bits, **8 bytes = 64 bits.**
And therefore, the minimum value is -2^{64-1} = -2^{63} = **-9,223,372,036,854,775,808**
The maximum value is $+2^{64-1} - 1$ = $+2^{63}$-1 = **+9,223,372,036,854,775,807**

Which means, any value between the range of -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 can be stored using a long type variable.

Therefore, distance between galaxy etc., can be stored using long data type.

Now let's look an example on long data type.

```
class Demo
{
	public static void main(String[] args)
	{
		long a;
		a = 9,223,372,036,854,775,807;
		System.out.println(a);
	}
}
```

Output:

```
Compilation error.
```

In Java, there is rule that whenever we want a value to be stored as long, at the end of the value a suffix of 'L' must be given.

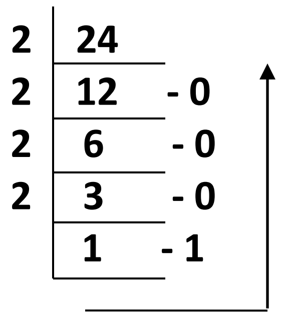
Hope now we have got the answer for our question of having four data types.

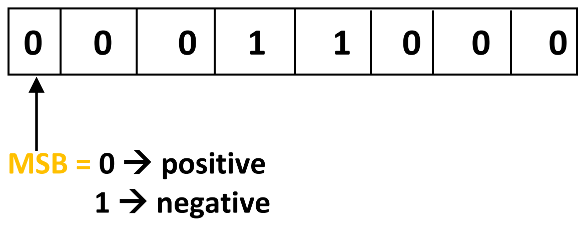
Integer values exists in different quantities, to handle different quantities of integer data different data types has been provided.

How Integer data is stored in the memory?

→The below is an example of how positive number is stored.

byte a = 24;





This type of conversion is called base-2 format.

→The below is an example of how negative number is stored.

byte a = -24;

For this, we require base-2 format + 2’s compliment of a number

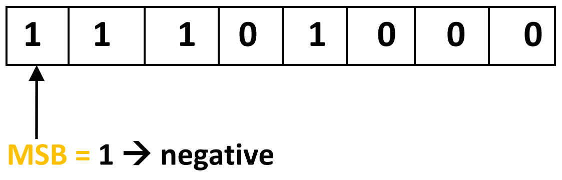
We know the base-2 value of 24 is **0001100.**

Now, let’s find the 2’s compliment of the number. **0 0 0 1 1 0 0 0**

Step-1: Find 1’s compliment of the number
by converting 0’s to 1’s and vice versa. **1 1 1 0 0 1 1 1**

Step-2: Find 2’s compliment by adding 1 to
the number. **+ 1**

This is the 2’s compliment of the number→ **1 1 1 0 1 0 0 0**

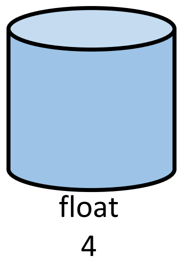


Real number type data:

In the real world we are rarely surrounded by Integer type data but often come across decimal numbers. For example, weight: 56.4kg, Temperature: 28.5 degrees, distance: Rajajinagar HO to metro station 0.5km, etc.

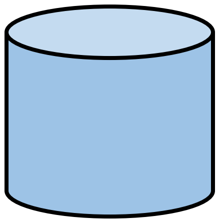
In Java to store such real number data types we have float, double.

1) **float** The float data type is a single-precision **32-bit** IEEE 754 floating-point. Float data type in Java **stores a decimal value** with **6-7 total digits of precision**. So for example 12.12345 can be saved as a float, but 12.123456789 cannot be saved as the float. When representing a float data type in Java we should **append the letter f** to the end of the data type, otherwise, it will save as double.
Assume, if we create an float type variable as **float a;**
It means in the memory **four bytes** is allocated and it is referred as **a**.



Range: **-34e^{38} to 34e^{38}**

2) double The double data type is a double-precision **64-bit** IEEE 754 floating-point. The double data type is normally the **default choice for decimal values**. The data type should never be used for precise values, such as currency. Double data type stores decimal values with **15-16 digits of precision**. The default value is 0.0d, this means that if you do not append f or d to the end of the decimal, the value will be stored as a double in Java.
Assume, if we create an double type variable as **double a;**
It means in the memory **eight bytes** is allocated and it is referred as **a**.



double
8

Range: **$-1.7e^{308}$ to $1.7e^{308}$**

Now let's see few examples to understand the above topic.

Example-1:

```
class Demo
{
        public static void main(String[] args)
        {
                double a=45.5;
                System.out.println(a);
        }
}
```

Output:

```
45.5
```

Example-2:

```
class Demo
{
        public static void main(String[] args)
        {
                float a=45.5;
                System.out.println(a);
        }
}
```

Output:

```
Compilation error.
```

In Java, there is rule that whenever we want a value to be stored as float, at the end of the value a suffix 'f' must be given, otherwise it is considered to be a double number.

Solution:

Method 1

```
class Demo
{
	public static void main(String[] args)
	{
		double a=(float)45.5;
		System.out.println(a);
	}
}
```

Method 2

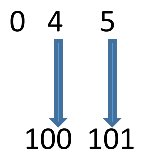
```
class Demo
{
	public static void main(String[] args)
	{
		double a=45.5f;
		System.out.println(a);
	}
}
```

Literals: In java values are called as literals.

Now let's see few examples to understand literals.

1) int a=45; ←Decimal literal
   S.O.P (a); // Output is 45.

2) int a=045; ←Octal literal
   S.O.P (a); // Output is 37.
   Note: Here 0 acts as the prefix to the literal and converts it from decimal to octal literal.



Now we have 100101 as our number, wherever 1 is present we take 2^{index} and add all.
We get **$2^0+2^2+2^5 = 1+4+32 = 37$**

3) int a=0x45;
   S.O.P (a); //Output is 69.
   Note: Here 0x acts as the prefix to the literal and converts it from decimal to hexadecimal literal.

$$\begin{tikzcd}
0 & 4 \arrow[d] & 5 \arrow[d] \\
 & 0100 & 0101
\end{tikzcd}$$

Now we have 01000101 as our number, wherever 1 is present we take 2^{index} and add all.
We get **$2^0+2^2+2^6$ = 1+4+64 = 69**

4) int a= 0b100101;
   S.O.P (a); // Output is 37.
   Note: Here 0b is a prefix for binary literals.

Wherever 1 is present we take 2^{index} and add all. As seen previously we'll get 37.